2015-3044

22nd AIAA Computational Fluid Dynamics Conference, June 24, 2014, Dallas, TX

General element shapes within a tensor-product higher-order space-time discontinuous-Galerkin formulation

Laslo T. Diosady* and Scott M. Murman†

NASA Ames Research Center, Moffett Field, CA, USA

A tensor-product higher-order space-time discontinuous-Galerkin method is extended to unstructured element shapes. The use of a tensor-product formulation is key to maintaining efficiency at high polynomial orders. The discrete system of equations arising at each space-time slab is solved using a Jacobian-free Newton-Krylov scheme. An alternating-direction-implicit (ADI) preconditioner for hexahedra is extended to prisms, pyramids and tetrahedra by solving on the tensor-product space corresponding to the quadrature points on the reference cube. Numerical results demonstrate the ADI preconditioner is able to reduce the stiffness associated with high polynomial orders for a scalar advection problem. A diagonalized variant of the ADI preconditioner for the compressible Navier-Stokes equations is used to perform simulations of the Taylor-Green vortex problem. Numerical results demonstrate the efficiency of higher-order methods for the simulation of compressible turbulent flows.

I. Introduction

Higher-order methods show potential for simulations requiring high spatial and temporal resolution, allowing for solutions with fewer degrees of freedom and lower computational cost to achieve the same error level as traditional second-order CFD methods.[1] In this work, we use a space-time discontinuous-Galerkin (DG) finite-element method, which extends to arbitrary order of accuracy in both space and time. Higher-order DG methods have been widely used for the solution of the compressible Euler and Navier-Stokes equations.[2–5] These methods are particularly attractive due to the possibility of using local h- and p-adaptation. In particular, the use of a space-time formulation allows for local adaptation in both the spatial and temporal directions, potentially leading to a significant reduction in cost as the increased resolution in time is only applied where necessary.

In our previous work[6–9] we have presented an efficient entropy-stable space-time discontinuous Galerkin method for the direct numerical simulation of compressible turbulent flows. The development of this numerical method has relied heavily upon a tensor-product formulation in order to maintain efficiency at high polynomial order. Thus, we have initially considered a formulation involving hexahedral elements. This limits the meshing flexibility of our tool when considering complex geometry. The use of more general element shapes (hexahedra, triangular prisms, pyramids and tetrahedra) allows for simulations of more complicated geometries. In this work we extend our hexahedral formulation to prisms, pyramids and tetrahedral elements. We envision a grid topology with predominantly hexahedral elements, with a small number of prisms, pyramids and tetrahedra. However, even with a relatively small number of non-hexahedral elements a tensor-product formulation is still necessary in order to maintain efficiency at high polynomial order.

In this work follow the unstructured spectral-element formulation of Karniadakis and Sherwin,[10] which has previously been applied to turbulent flow simulations.[10–12] However, unlike this previous work which used explicit time-stepping, our space-time discontinuous-Galerkin scheme requires the solution of a globally coupled system of equations for each space-time slab. As higher-order methods have increased stiffness

*Science and Technology Corp, laslo.diosady@nasa.gov

†Scott.M.Murman@nasa.gov

relative to traditional second-order methods, efficient preconditioning techniques are required to solve this system of equations. In our previous work we have presented preconditioners which take advantage of the tensor-product formulation on hexahedral elements enabling us to overcome the increased stiffness associated with high-order.[8] In this paper we consider extensions of these preconditioners to prisms, pyramids and tetrahedra. Numerical results demonstrate that the resulting convergence rates are independent of solution order for a fixed number of degrees of freedom.

This paper is organized as follows. In Section II we present our numerical method including the tensor-product bases employed on the unstructured grid and efficient evaluation using the sum-factorization approach. Section III presents an alternating direction implicit (ADI) preconditioner for a scalar advection equation which takes advantage of our tensor-product formulation. Section IV presents the extension of the ADI preconditioner for the solution of the compressible Navier-Stokes equations. In Section V we apply our numerical method to the solution of compressible flows. Finally, we provide a summary and conclusions in Section VI.

II. Numerical Method

The compressible Navier-Stokes equations are written in conservative form as:

$$\boldsymbol{u}_{,t} + \nabla \cdot (\boldsymbol{f}^I - \boldsymbol{f}^V) = 0 \tag{1}$$

where $(\cdot)_{,t}$ denotes partial differentiation with respect to time. The conservative state vector is

$$\boldsymbol{u} = \begin{bmatrix} \rho \\ \rho \boldsymbol{V} \\ \rho E \end{bmatrix}, \tag{2}$$

where ρ is the density, $\boldsymbol{V}$ is the velocity, and E the total energy. The inviscid and viscous fluxes are given, respectively, by:

$$\boldsymbol{f}^I = \begin{bmatrix} \rho \boldsymbol{V} \\ \rho \boldsymbol{V}\boldsymbol{V}^T + p\boldsymbol{I} \\ \rho \boldsymbol{V} H \end{bmatrix}, \qquad \text{and} \qquad \boldsymbol{f}^V = \begin{bmatrix} 0 \\ \boldsymbol{\tau} \\ \boldsymbol{\tau}\boldsymbol{V} - \kappa_T \nabla T \end{bmatrix}, \tag{3}$$

where p is the static pressure, $H = E + \frac{p}{\rho}$ is the total enthalpy, $\boldsymbol{\tau}$ the viscous stress tensor, κ_T is the thermal conductivity, $T = p/\rho R$ is the temperature, and R is the gas constant. The pressure is given by:

$$p = (\gamma - 1)\left(\rho E - \tfrac{1}{2}\rho \boldsymbol{V}^2\right), \tag{4}$$

where γ is the specific heat ratio. The viscous stress tensor, $\boldsymbol{\tau}$, is given by:

$$\boldsymbol{\tau} = \mu\left(\nabla \boldsymbol{V} + \nabla \boldsymbol{V}^T\right) - \lambda(\nabla \cdot \boldsymbol{V})\boldsymbol{I} \tag{5}$$

where μ is the viscosity, $\lambda = \frac{2}{3}\mu$ is the bulk viscosity.

Applying a change of variables $\boldsymbol{u} = \boldsymbol{u}(\mathbf{v})$, where $\mathbf{v}$ are the entropy variables:

$$\mathbf{v} = \begin{bmatrix} -\frac{s}{\gamma-1} + \frac{\gamma+1}{\gamma-1} + \frac{\rho E}{p} \\ \frac{\rho \boldsymbol{V}}{p} \\ -\frac{\rho}{p} \end{bmatrix} \tag{6}$$

we rewrite the Navier-Stokes equations as:

$$\boldsymbol{A}_0 \mathbf{v}_{,t} + \bar{\boldsymbol{A}} \nabla \mathbf{v} - \nabla \cdot (\bar{\bar{\boldsymbol{K}}} \nabla \mathbf{v}) \quad = \quad 0 \tag{7}$$

with symmetric $\boldsymbol{A}_0 = \boldsymbol{u}_{,\mathbf{v}}$, $\bar{\boldsymbol{A}} = \boldsymbol{f}^I_{,\boldsymbol{u}} \boldsymbol{A}_0 = \boldsymbol{f}^I_{,\mathbf{v}}$ and $\bar{\bar{\boldsymbol{K}}} = \boldsymbol{f}^V_{,\nabla \boldsymbol{u}} \boldsymbol{A}_0 = \boldsymbol{f}^V_{,\nabla \mathbf{v}}$.[13]

We proceed to discretize (7) as follows. The domain, Ω, is partitioned into non-overlapping elements, κ, while the time is partitioned into intervals (time-slabs), $I^n = [t^n, t^{n+1}]$. Define $\mathcal{V}_h = \left\{\boldsymbol{w}, \boldsymbol{w}|_{\kappa \times I} \in [\mathcal{P}(\kappa \times I)]^{Rank}\right\}$,

the space-time finite-element space consisting of piece-wise polynomial functions in both space and time on each element, where $Rank = 5$ is the number of conservation equations. We seek a solution $\mathbf{v} \in \mathcal{V}_h$ such that the weak form:

$$\begin{aligned} \boldsymbol{r}(\boldsymbol{w},\mathbf{v}) &= \sum_{\kappa} \Big\{ \int_I \int_\kappa - \Big(\boldsymbol{w}_{,t} \cdot \boldsymbol{u} + \nabla \boldsymbol{w} \cdot (\boldsymbol{f}^I - \boldsymbol{f}^V) \Big) \\ &\quad + \int_I \int_{\partial\kappa} \boldsymbol{w} \cdot (\widehat{\boldsymbol{f}^I \cdot \boldsymbol{n}} - \widehat{\boldsymbol{f}^V \cdot \boldsymbol{n}}) \\ &\quad + \int_\kappa \boldsymbol{w}(t_-^{n+1}) \cdot \boldsymbol{u}(t_-^{n+1}) - \boldsymbol{w}(t_+^n) \cdot \boldsymbol{u}(t_-^n) \Big\} = 0 \end{aligned} \tag{8}$$

is satisfied for all $\boldsymbol{w} \in \mathcal{V}_h$, where $\boldsymbol{u} = \boldsymbol{u}(\mathbf{v})$ as given above. Here $\widehat{\boldsymbol{f}^I \cdot \boldsymbol{n}}$ and $\widehat{\boldsymbol{f}^V \cdot \boldsymbol{n}}$ denote numerical flux functions approximating the inviscid and viscous fluxes, respectively, while $\boldsymbol{n}$ is the outward pointing normal vector. In this work, the inviscid flux is discretized using the method of Ismail and Roe,[14] while the viscous flux is discretized using an interior penalty method.

We use a tensor-product basis such that on each element $\mathbf{v}$ is given by

$$\mathbf{v}(\mathbf{x}(\eta), t(\tau)) = \mathbf{v}_{ijkl} \Phi_{ijkl} \tag{9}$$

where Φ_{ijkl} are the set of basis functions which are orthogonal on the reference element, while $\mathbf{v}_{ijkl}$ are the coefficients. We use a combination of Lagrange and Dubiner basis functions:[10]

$$\Phi^{hex}_{ijkl} = \phi_i(\eta_1)\phi_j(\eta_2)\phi_k(\eta_3)\phi_l(\tau) \qquad 0 \le i,j,k,l < N \tag{10}$$

$$\Phi^{prism}_{ijkl} = \psi^b_{ij}(\eta_1)\psi^a_j(\eta_2)\phi_k(\eta_3)\phi_l(\tau) \qquad 0 \le i+j < N, \qquad 0 \le k,l < N \tag{11}$$

$$\Phi^{pyramid}_{ijkl} = \psi^c_{ijk}(\eta_1)\psi^a_j(\eta_2)\psi^a_k(\eta_3)\phi_l(\tau) \qquad 0 \le i + \max(j,k) < N, \qquad 0 \le l < N \tag{12}$$

$$\Phi^{tet}_{ijkl} = \psi^c_{ijk}(\eta_1)\psi^b_{jk}(\eta_2)\psi^a_k(\eta_3)\phi_l(\tau) \qquad 0 \le i+j+k < N, \qquad 0 \le l < N \tag{13}$$

where $\mathbf{x}(\eta)$ defines a mapping from a reference cube, $\eta \subset [-1,1]^3$, to physical space, while $t(\tau)$ is the mapping from the reference interval $[-1,1]$ to the time interval $[t^n, t^{n+1}]$. Here, N is the solution order, ϕ_i are one-dimensional Lagrange basis functions defined at Gauss-Legendre points, while the Dubiner basis functions ψ^a_k, ψ^b_{jk} and ψ^c_{ijk} are given by:

$$\psi^a_k(\eta) = P^{0,0}_k(\eta) \qquad \psi^b_{jk}(\eta) = \left(\tfrac{1-\eta}{2}\right)^k P^{2k+1,0}_j(\eta) \qquad \psi^c_{ijk}(\eta) = \left(\tfrac{1-\eta}{2}\right)^{j+k} P^{2j+2k+1,0}_i(\eta) \tag{14}$$

where $P^{\alpha,\beta}_p$ denotes the pth-order Jacobi polynomial.[10] We note that for the hexahedral elements, the coefficients $\mathbf{v}_{ijkl}$ correspond to nodal values, while this is not generally the case for the other element types, where modal bases are used.

The integrals in (8) are evaluated using numerical quadrature. For example:

$$\begin{aligned} &\frac{2}{\Delta t} \int_I \int_\kappa - \Big(\boldsymbol{w}_{,t} \cdot \boldsymbol{u} + \nabla \boldsymbol{w} \cdot (\boldsymbol{f}^I - \boldsymbol{f}^V)) \Big) \\ \simeq\ &\Big\{ - \Big(\tau_{,t} \boldsymbol{w}_{,\tau} \cdot \boldsymbol{u} + \nabla_\eta \boldsymbol{w} \cdot (\tilde{\boldsymbol{f}}^I - \tilde{\boldsymbol{f}}^V)) \Big) |\mathbf{x}_{,\eta}| \Big\}_{\eta_p \eta_q \eta_r \tau_s} w_p w_q w_r w_s \end{aligned} \tag{15}$$

where $\eta_p, \eta_q, \eta_r, \tau_s$ are one-dimensional Gauss-Jacobi quadrature points, and w_p, w_q, w_r and w_t are the associated quadrature weights. $|\mathbf{x}_{,\eta}|$ denotes the Jacobian of the mapping from element reference cube to physical space, ∇_η denotes differentiation with respect to the reference coordinate η, while $\tilde{\boldsymbol{f}}^I = \eta_{,\mathbf{x}} \boldsymbol{f}^I$ and $\tilde{\boldsymbol{f}}^V = \eta_{,\mathbf{x}} \boldsymbol{f}^V$ are the fluxes mapped to the local element coordinate system. In this work we use a quadrature rule with twice as many quadrature points as nodal points in order to reduce the quadrature error (we ensure exact integration of cubic nonlinearities) thereby improving the nonlinear stability of our scheme.[6,12]

The remaining integrals appearing in (8) are evaluated in a similar manner, which may be described as a sequence of three steps:

1. Evaluate the state ($\mathbf{v}$) and gradient ($\nabla_\eta \mathbf{v}$) at the quadrature points.

2. Evaluate the fluxes ($\tilde{\boldsymbol{f}}^I$ and $\tilde{\boldsymbol{f}}^V$) at the quadrature points.

3. Multiply the fluxes with the basis functions ($\boldsymbol{w}$) or gradients ($\nabla_\eta \boldsymbol{w}$).

A key requirement for efficiency at high polynomial order is the evaluation of the first and third steps using the sum-factorization approach,[6,15] which allows the multiplication of the basis functions to be performed as a sequence of one-dimensional operations. For example the evaluation of the state at a quadrature point in a tetrahedron may be written as:

$$\begin{aligned} \mathbf{v}_{pqrs} &= \sum_{ijkl} \Phi_{ijkl}|_{pqrs} \mathbf{v}_{ijkl} && (16) \\ &= \sum_{l}\sum_{k}\sum_{j}\sum_{i} \phi_l|_s \psi_k|_r \psi_{jk}|_q \psi_{ijk}|_p \mathbf{v}_{ijkl} && (17) \\ &= \sum_{l} \phi_l|_s \left(\sum_{k} \psi_k|_r \left(\sum_{j} \psi_{jk}|_q \left(\sum_{i} \psi_{ijk}|_p \mathbf{v}_{ijkl} \right) \right) \right) && (18) \end{aligned}$$

A similar operation may be performed when multiplying by the basis functions in order to form the residual. This results in a residual evaluation cost which scales as $O(N^{d+1})$ for each space-time element where N is the solution order while d is the number of spatial-temporal dimensions (for unsteady 3D simulations $d = 4$). Thus, for a fixed number of spatial-temporal degrees of freedom the residual evaluation scales linearly with the solution order. However, for moderate solution orders, $N = 4 - 16$, we can offset this linear scaling by using optimized numerical kernels.[6]

In our hexahedral formulation we have made use of further simplifications in order to increase the efficiency of our numerical scheme. While we compute the residual using a rule with twice the number of quadrature points as solution points, we employ a lower order quadrature rule when computing the linearized residual in our solution procedure. In particular we evaluate the linearized residual using a quadrature rule of the same order as the solution. As we have noted previously, the basis function coefficients for hexahedral elements correspond to the nodal values at the Gauss-Legendre points. Thus, for evaluating the linearized residual we employ a collocated quadrature rule, where the nodal points are used as quadrature points. When using a collocated quadrature rule, the solution value at the nodes is directly available, while the gradient of the solution may be obtained using a summation in only a single coordinate direction. Namely:

$$\left.\frac{\partial \mathbf{v}}{\partial \eta_1}\right|_{pjkl} = \sum_i \left.\frac{\partial \phi_i}{\partial \eta_1}\right|_p \mathbf{v}_{ijkl} \qquad \left.\frac{\partial \mathbf{v}}{\partial \eta_2}\right|_{iqkl} = \sum_j \left.\frac{\partial \phi_j}{\partial \eta_1}\right|_q \mathbf{v}_{ijkl} \tag{19}$$

$$\left.\frac{\partial \mathbf{v}}{\partial \eta_3}\right|_{ijrl} = \sum_k \left.\frac{\partial \phi_k}{\partial \eta_1}\right|_r \mathbf{v}_{ijkl} \qquad \left.\frac{\partial \mathbf{v}}{\partial \tau}\right|_{ijks} = \sum_l \left.\frac{\partial \phi_l}{\partial \tau}\right|_s \mathbf{v}_{ijkl} \tag{20}$$

This reduces the number of operations to evaluate the state at the quadrature points by a factor of approximately 5 relative to a naive approach. For prisms, pyramids, and tetrahedra the coefficients $\mathbf{v}_{ijkl}$ no longer correspond to nodal values. Thus, evaluating the nodal values for a quadrature rule with the same order as the solution generally involves the coefficients of all basis functions corresponding to the same time level (we still take advantage of a collocated rule in the temporal direction). The nodal values are evaluated as given by:

$$\mathbf{v}|_{pqrl} = \sum_k \psi_k|_r \left(\sum_j \psi_{jk}|_q \left(\sum_i \psi_{ijk}|_p \mathbf{v}_{ijkl} \right) \right) \tag{21}$$

The derivative of the basis functions with respect to η_i take on a similar form, while the derivative with respect to τ also involves a sum over the temporal direction. Thus, evaluating the basis functions and derivatives at all quadrature points for pyramids and tetrahedra is approximately 1.5 times the number of operations as compared with a collocated quadrature approach on the hexahedra. For prisms, a nodal basis is used in the directions which are not collapsed, thus a collocated quadrature rule may be used in these directions, leading to an operation count that is also roughly 1.5 times that as compared with hexahedra. These estimates are verified by numerical experiments. The CPU time to perform gradient and state evaluations for the different

element shapes normalized by the CPU time for a hexahedron are presented in Figure 1. The CPU time is somewhat larger than that predicted by the analysis which may be due to improved cache efficiency on the hexahedra relative to the other element shapes.

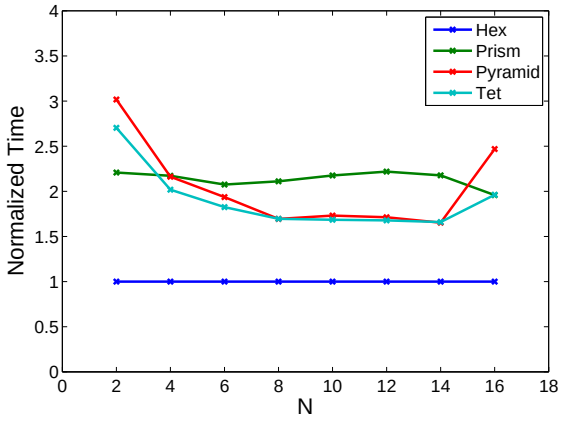


Figure 1. CPU time for state evaluation at "collocated" quadrature points on an element, normalized by CPU time for a hexahedron.

Next we consider the evaluation of the state and its gradient on the faces of elements, required for the integrals on the second line of (8). On the hexahedra, it is convenient to consider this as a sequence of two operations:

1. Evaluate the state and normal gradients at a set of nodal points on the element face.
2. Evaluate the state and gradient at the face quadrature points based on the face nodal points.

When performing parallel simulations, communication between processors sharing an element face involves only face nodal and normal gradient values as opposed to either entire element data or face quadrature point data, thereby minimizing the total amount of communication. Specifically, evaluation of the face nodal and normal gradient values involves only a simple sum in the direction normal to a face:

$$\mathbf{v}|_{\pm jkl} = \sum_i \phi_i|_{\pm 1}\mathbf{v}_{ijkl} \qquad \mathbf{v}|_{i\pm kl} = \sum_j \phi_j|_{\pm 1}\mathbf{v}_{ijkl} \qquad \mathbf{v}|_{ij\pm l} = \sum_k \phi_k|_{\pm 1}\mathbf{v}_{ijkl} \tag{22}$$

$$\left.\frac{\partial \mathbf{v}}{\partial \eta_1}\right|_{\pm jkl} = \sum_i \left.\frac{\partial \phi_i}{\partial \eta_1}\right|_{\pm 1}\mathbf{v}_{ijkl} \qquad \left.\frac{\partial \mathbf{v}}{\partial \eta_1}\right|_{i\pm kl} = \sum_j \left.\frac{\partial \phi_j}{\partial \eta_2}\right|_{\pm 1}\mathbf{v}_{ijkl} \qquad \left.\frac{\partial \mathbf{v}}{\partial \eta_1}\right|_{ij\pm l} = \sum_k \left.\frac{\partial \phi_k}{\partial \eta_3}\right|_{\pm 1}\mathbf{v}_{ijkl} \tag{23}$$

The operation count for the evaluation of the face nodal values scales as $O(N^d)$, while evaluating the state at the face quadrature points using the face nodal values involves a sum factorization of dimension $d-1$, again leading to a cost which scales as $O(N^d)$.

We wish to follow a similar procedure as outlined above for evaluating the state and gradient on faces of prisms, pyramids, and tetrahedra. In particular, we will evaluate the state and normal gradients on a set of nodal points on the (potentially collapsed) reference quadrilateral corresponding to each face. Then we use these nodal values to compute the state and gradient at the face quadrature points. We note that the basis functions are polynomial in the space of the reference quadrilateral, whether or not the face corresponds to a triangular or quadrilateral surface. The polynomial space corresponding to the element basis projected onto the face is a subset of the space spanned by the tensor product of Lagrange polynomials on the reference quadrilateral. Thus, the sum factorization approach identical to that used for the faces of the hexahedra may be used once the solution is available at nodal points on the face. It remains to evaluate the solution at the face nodal points given the solution coefficients.

As mentioned previously, the evaluation of the solution at nodal points on prisms, pyramids, and tetra-

hedra generally involves all solution points. We evaluate the solution on the faces of a tetrahedron as:

$$\mathbf{v}|_{-qrl} = \sum_k \psi_k|_r \left(\sum_j \psi_{jk}|_q \left(\sum_i \psi_{ijk}|_- \mathbf{v}_{ijkl} \right) \right) \tag{24}$$

$$\mathbf{v}|_{p-rl} = \sum_k \psi_k|_r \left(\sum_j \psi_{jk}|_- \left(\sum_i \psi_{ijk}|_p \mathbf{v}_{ijkl} \right) \right) \tag{25}$$

$$\mathbf{v}|_{pq\pm l} = \sum_k \psi_k|_\pm \left(\sum_j \psi_{jk}|_q \left(\sum_i \psi_{ijk}|_p \mathbf{v}_{ijkl} \right) \right) \tag{26}$$

where the four faces of the tetrahedron correspond to $\eta_1 = -1$, $\eta_2 = -1$, $\eta_3 = -1$ and $\eta_3 = +1$. The evaluation of the face nodal values scales as $O(N^3)$ for the face $\eta_1 = -1$, which is similar to that for the faces of the hexahedron. However, for the remaining faces of the tetrahedron the cost scales as $O(N^4)$ since the innermost sum involves all degrees of freedom. This implies that the cost of the face evaluation will be more significant for tetrahedra than for hexahedra. A similar situation also occurs with the faces of the pyramids and prisms.

Once again we perform numerical simulations to confirm the behavior of this analysis. Figure 2 shows the cost of evaluating the state and normal gradient at the face nodal points of hexahedra, prisms, pyramids, and tetrahedra. For each element shape we compute the minimum and maximum CPU time over the different faces (normalized by the average CPU time for the face of a hexahedron). We note that for the hexahedron the mininum and maximum CPU time vary by about a factor of two, where the variation is due to slightly improved cache efficiency for particular faces. For the other element shapes, the minimum CPU time relative to the hexahedron is roughly constant, while the maximum increases linearly with solution order as expected by the analysis.

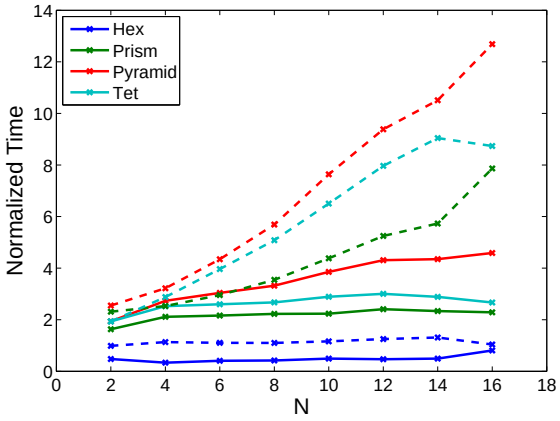


Figure 2. CPU time for state evaluation at face nodal points, normalized by average CPU time for a hexahedron. (Dashed line corresponds to most expensive face, while solid line corresponds to least expensive face.)

III. Preconditioning: Scalar-Advection

Equation (8) represents a globally coupled system of nonlinear equations which need to be solved for each time-slab. In this work we use a Jacobian-free Newton-Krylov method describe in detail in our previous papers.[6,8] Preconditioning is necessary to overcome the increased stiffness associated with high-order. In our previous work, we have develop preconditioners with memory requirements no larger than that required for residual evaluations. In particular, we develop element-wise block-Jacobi preconditioners where the elemental blocks are solved approximately, taking advantage of the tensor-product formulation of our finite-element scheme. We employ either of two preconditioners: an Alternating-Direction-Implicit (ADI)[16] preconditioner or a preconditioner based on the Fast Diagonalization Method (FDM).[17] These preconditioners cannot be directly applied for prism, pyramids, and must be extended to handle these element types.

We describe the formulation of our preconditioners by considering a steady constant-coefficient linear advection problem in two dimensions:

$$\boldsymbol{a} \cdot \nabla v \quad = \quad f \tag{27}$$

where v is the conserved scalar, $\boldsymbol{a}$ is a divergence-free velocity field, and f is a forcing term.

Applying our discontinuous-Galerkin discretization using an upwind-flux we obtain an elemental block Jacobian which corresponds to the following operator:

$$r(w|_\kappa, v|_\kappa) \quad = \quad -\int_\kappa (\nabla w \cdot \boldsymbol{a} v) + \int_{\partial\kappa} a_n^+ wv \tag{28}$$

where $a_n = \boldsymbol{a} \cdot \boldsymbol{n}$, $a_n^\pm = \frac{1}{2}(a_n \pm |a_n|)$, and $\nabla_{\boldsymbol{n}}$ denotes differentiation in the normal direction. Mapping the physical element to the reference square, the elemental block Jacobian for a single element may be written as:

$$\begin{aligned} r(w|_\kappa, v|_\kappa) \quad = \quad & + \int_{\eta_2} \left(- \int_{\eta_1} (|\mathbf{x}_{,\eta}| \tilde{a}_1 w_{,\eta_1} v) + \left[|\mathbf{x}_{,\eta}| \tilde{a}_1^+ wv \right]_{\eta_1=-1}^{\eta_1=1} \right) \\ & + \int_{\eta_1} \left(- \int_{\eta_2} (|\mathbf{x}_{,\eta}| \tilde{a}_2 w_{,\eta_2} v) + \left[|\mathbf{x}_{,\eta}| \tilde{a}_2^+ wv \right]_{\eta_2=-1}^{\eta_2=1} \right) \end{aligned} \tag{29}$$

where $\tilde{a}_i = \eta_{i,x_j} a_j$, $\tilde{a}_i^+ = \left\{ \frac{1}{2}(\tilde{a}_i + |\tilde{a}_i|) \text{ if } \eta_i = 1 \text{ and } \frac{1}{2}(\tilde{a}_i - |\tilde{a}_i|) \text{ if } \eta_i = -1 \right\}$ and $|\mathbf{x}_{,\eta}|$ is the Jacobian of the mapping from the reference square to the physical element. It is convenient to factor the Jacobian of the mapping, $|\mathbf{x}_{,\eta}|$, into two terms $|\mathbf{x}_{,\eta}| = |\mathbf{x}_{,\xi}||\xi_{,\eta}|$, where ξ are the reference coordinates in the element (as opposed to the reference square). Assuming the mapping, $\mathbf{x}(\xi)$, from the reference element to the physical space is constant the elemental block Jacobian may be written as:

$$\begin{aligned} \frac{1}{|\mathbf{x}_\xi|} r(w|_\kappa, v|_\kappa) \quad = \quad & + \int_{\eta_2} \left(- \int_{\eta_1} (|\xi_{,\eta}| \eta_{1,\xi_k} \bar{a}_k w_{,\eta_1} v) + \left[|\xi_{,\eta}| (\eta_{1,\xi_k} \bar{a}_k)^+ wv \right]_{\eta_1=-1}^{\eta_1=1} \right) \\ & + \int_{\eta_1} \left(- \int_{\eta_2} (|\xi_{,\eta}| \eta_{2,\xi_k} \bar{a}_k w_{,\eta_2} v) + \left[|\xi_{,\eta}| (\eta_{2,\xi_k} \bar{a}_k)^+ wv \right]_{\eta_2=-1}^{\eta_2=1} \right) \end{aligned} \tag{30}$$

where $\bar{a}_i = \xi_{i,x_j} a_j$, and also $\tilde{a}_i = \eta_{i,\xi_j} \bar{a}_j$. We note that $\bar{a}_i$ are constant on an element, while $\tilde{a}_i$ depend upon the mapping from the reference square to the reference element. For quadrilateral elements (or for hexahedral elements in 3D) $\xi_{,\eta}$ is the identity matrix and $\bar{a}_i = \tilde{a}_i$.

We first consider only quadrilateral elements. We recognize that each line on the right-hand side of (29) corresponds to an advection operator along an axis of our reference element. Employing the tensor-product basis, $v = v_1(\eta_1)v_2(\eta_2)$, $w = w_1(\eta_1)w_2(\eta_2)$ the elemental block Jacobian is rewritten conveniently as:

$$\begin{aligned} \frac{1}{|\mathbf{x}_\xi|} r(w|_\kappa, v|_\kappa) \quad = \quad & + \left(- \int_{\eta_1} (|\xi_{,\eta}| \eta_{1,\xi_k} \bar{a}_k w_{1,\eta_1} v_1) + \left[|\xi_{,\eta}| (\eta_{1,\xi_k} \bar{a}_k)^+ w_1 v_1 \right]_{\eta_1=-1}^{\eta_1=1} \right) \int_{\eta_2} w_2 v_2 \\ & + \int_{\eta_1} w_1 v_1 \left(- \int_{\eta_2} (|\xi_{,\eta}| \eta_{2,\xi_k} \bar{a}_k w_{2,\eta_2} v_2) + \left[|\xi_{,\eta}| (\eta_{2,\xi_k} \bar{a}_k)^+ w_2 v_2 \right]_{\eta_2=-1}^{\eta_2=1} \right), \end{aligned} \tag{31}$$

which corresponds to the discrete system:

$$A_\kappa^{quad} \quad = \quad |\mathbf{x}_\xi| \left((D_1 \otimes M_2) + (M_1 \otimes D_2) \right) \tag{32}$$

where M_i are one-dimensional mass matrices, whose $[m, n]$ entry is given by:

$$M_i[m, n] \quad = \quad \int_{\eta_i} w_i v_i \tag{33}$$

while D_i are one-dimensional advection operators:

$$D_i[m, n] \quad = \quad -\int_{\eta_i} (\bar{a}_i w_{i,\eta_i} v_i) + \left[\bar{a}_i^+ w_i v_i \right]_{\eta_i=-1}^{\eta_i=1} \tag{34}$$

where v_i and w_i correspond, respectively, to m and n. Factoring out the elemental mass matrix gives:

$$A_\kappa^{quad} = |\mathbf{x}_\xi|(M_1 \otimes M_2)\{(\tilde{D}_1 \otimes I) + (I \otimes \tilde{D}_2)\} \tag{35}$$

where $\tilde{D}_i = M_i^{-1} D_i$ while I denotes the identity matrix. We seek to efficiently compute an inverse of A_κ^{quad}. We note that the term $M_\kappa \equiv |\mathbf{x}_\xi|(M_1 \otimes M_2)$ appearing in (35) is simply the elemental mass matrix, which is diagonal and thus easily inverted. It remains to invert the matrix:

$$(\tilde{D}_1 \otimes I) + (I \otimes \tilde{D}_2) \tag{36}$$

We have previously developed an alternating direction implicit (ADI) preconditioner and a preconditioner based on the fast-diagonalization method (FDM) to approximately invert (36). These preconditioners take advantage of the fact that (36) is in separable form. When developing preconditioners for general element shapes, we wish to form similar tensor-product systems. We consider first the triangular element and then present a generalization for three-dimensional elements.

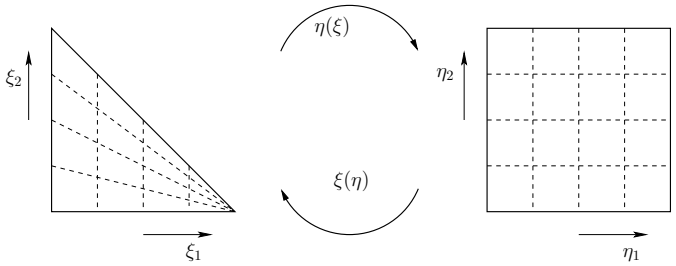


Figure 3. Mapping from reference triangle to reference square

Figure 3 depicts the mapping from the reference triangle to the reference square. Specifically, the mapping is given by:

$$\eta_\xi = \begin{bmatrix} 1 & 0 \\ \frac{1+\eta_2}{1-\eta_1} & \frac{2}{1-\eta_1} \end{bmatrix} \qquad \text{and} \qquad |\xi_\eta| = \tfrac{1-\eta_1}{2} \tag{37}$$

where we note that the mapping is singular at the collapsed corner of the triangle corresponding to $\xi_1 = \eta_1 = 1$. The elemental block Jacobian is given by:

$$\begin{aligned} \tfrac{1}{|\mathbf{x}_\xi|} r(w|_\kappa, v|_\kappa) = & + \int_{\eta_2} \left(- \int_{\eta_1} \left(\tfrac{1-\eta_1}{2} \bar{a}_1 w_{,\eta_1} v \right) + \left[\tfrac{1-\eta_1}{2} \bar{a}_1^+ wv \right]_{\eta_1=-1} \right) \\ & + \int_{\eta_1} \left(- \int_{\eta_2} \left(\left(\tfrac{1+\eta_2}{2} \bar{a}_1 + \bar{a}_2 \right) w_{,\eta_2} v \right) + \left[\left(\tfrac{1+\eta_2}{2} \bar{a}_1 + \bar{a}_2 \right)^+ wv \right]_{\eta_2=-1}^{\eta_2=1} \right) \end{aligned} \tag{38}$$

As in the case of the quadrilateral, we recognize that each line in (38) corresponds to the discretization of an advection problem along a coordinate direction of the reference square. However, in the triangular case, the coefficient is spatially varying. We consider a basis formed by the tensor-product of Lagrange polynomials defined at the Gauss-Jacobi quadrature points on the reference quadrilateral. Unfortunately, the space defined by this basis includes functions which, when mapped to the reference triangle, are multi-valued at the corner of the triangle corresponding to the collapsed edge ($\eta_1 = 1$) of the quadrilateral. In particular, the space spanned by the Dubiner basis is a subset of this tensor-product space, constrained to ensure that all functions are polynomial on the triangle and single-valued at the collapsed edge. Our preconditioner for the triangle will be based on solving a local problem in the artificially constructed tensor-product space using our previously developed preconditioners, then restricting the solution to the space spanned by the Dubiner basis.

Consider the tensor-product space where $v = v_1(\eta_1)v_2(\eta_2)$ and $w = w_1(\eta_1)w_2(\eta_2)$. Since the polynomial space spanned by the Dubiner basis is a subspace of this artificially constructed tensor-product space, the mass-matrix and elemental block-Jacobian for the triangle may be recovered by a simple Galerkin projection:

$$M_\kappa^{tri} = \Phi^T M_{\kappa^\dagger}^{tri} \Phi \qquad \text{and} \qquad A_\kappa^{tri} = \Phi^T A_{\kappa^\dagger}^{tri} \Phi \tag{39}$$

where $M^{tri}_{\kappa^\dagger}$ is the diagonal matrix of quadrature weights (scaled by the Jacobian of the mapping from reference to physical space), $A^{tri}_{\kappa^\dagger}$ is the discrete system corresponding to (38) with the tensor-product basis, while Φ is the matrix of the Dubiner basis functions evaluated at the quadrature points. The matrix operations in (39) may be viewed simply as a restatement of the quadrature formula used to evaluate M^{tri}_{κ} and A^{tri}_{κ}. In particular, $M^{tri}_{\kappa^\dagger}$ has the form:

$$M^{tri}_{\kappa^\dagger} = |x_\xi|(M_{1^\dagger} \otimes M_2) \tag{40}$$

where

$$M_{1^\dagger}[m,n] = \int_{\eta_1=-1}^{1} \tfrac{1-\xi_1}{2} w_1 v_1 \tag{41}$$

Note that $M_{1^\dagger}$ and M_2 can be evaluated exactly using a Gauss-Jacobi quadrature which exactly integrates $\int_{-1}^{1}(1-\eta)^\alpha f(\eta)$ for polynomials functions $f(\eta)$ up to order $2N+1$. In particular we use a Gauss-Jacobi quadrature rule with $\alpha = 1$ in the η_1-direction and $\alpha = 0$ in the η_2-direction. $M_{1^\dagger}$ and M_2 thus correspond to diagonal matrices with diagonals given by the Gauss-Jacobi quadrature weights.

Similarly, $A^{tri}_{\kappa^\dagger}$ has the form:

$$\tilde{A}^{tri}_{\kappa^\dagger} = |\mathbf{x}_\xi| \left((D_{1^\dagger} \otimes M_2) + (N_{1^\dagger} \otimes D_2)\right) \tag{42}$$

where

$$D_{1^\dagger}[m,n] = -\int_{\eta_1} \left(\tfrac{1-\eta_1}{2} \bar{a}_1 w_{1,\eta_1} v_1\right) + \left[\tfrac{1-\eta_1}{2} \bar{a}_1^+ w_1 v_1\right]_{\eta_1=-1} \tag{43}$$

$$D_2[m,n] = -\int_{\eta_2} \left(\left(\tfrac{1+\eta_2}{2} \bar{a}_1 + \bar{a}_2\right) w_{2,\eta_2} v_2\right) + \left[(\tfrac{1+\eta_2}{2} \bar{a}_1 + \bar{a}_2)^+ w_2 v_2\right]_{\eta_2=-1}^{\eta_2=1} \tag{44}$$

$$N_{1^\dagger}[m,n] = \int_{\eta_1=-1}^{1} w_1 v_1 \tag{45}$$

Once again, we can evaluate the integrals appearing in (45) discretely using the same Gauss-Jacobi quadrature rules. In this case, the quadrature rule is not exact for $N_{1^\dagger}$, however we recover an approximation $\tilde{A}^{tri}_{\kappa^\dagger}$ which exposes the tensor-product formulation:

$$\tilde{A}^{tri}_{\kappa^\dagger} = |\mathbf{x}_\xi|(M_{1^\dagger} \otimes M_2)\left((\tilde{D}_{1^\dagger} \otimes I) + (G_{1^\dagger} \otimes \tilde{D}_2)\right) \tag{46}$$

where $G_{1^\dagger} = M_{1^\dagger}^{-1} N_{1^\dagger}$ is simply a diagonal matrix with diagonal entries given by $\frac{2}{1-\xi_1}$ evaluated at each quadrature point. In particular, we can write A^{tri}_{κ} as:

$$A^{tri}_{\kappa} = \Phi^T \tilde{A}^{tri}_{\kappa^\dagger} \Phi + E^{tri}_{\kappa} = \tilde{A}^{tri}_{\kappa} + E^{tri}_{\kappa} \tag{47}$$

where $\tilde{A}^{tri}_{\kappa} \equiv \Phi^T \tilde{A}^{tri}_{\kappa^\dagger} \Phi$ corresponds to the evaluation using a Gauss-Jacobi quadrature rule with $\alpha = 1$ on the edges corresponding to $\eta_2 = -1$ and $\eta_2 = 1$, while E^{tri}_{κ} is the correction corresponding to evaluating these integrals exactly using a quadrature rule with $\alpha = 0$. We note that the error term E^{tri}_{κ} is a simple rank-1 matrix of the form $E^{tri}_{\kappa} = e_1 e_2^T$. Thus if we can efficiently compute (or apply) $\tilde{A}^{tri^{-1}}_{\kappa}$ then we can compute (or apply) $A^{tri^{-1}}_{\kappa}$ efficiently using the Sherman-Morrison formula:

$$A^{tri^{-1}}_{\kappa} = \left(I - \frac{\tilde{A}^{tri^{-1}}_{\kappa} e_1 e_2^T}{1 + e_2^T \tilde{A}^{tri^{-1}}_{\kappa} e_1}\right) \tilde{A}^{tri^{-1}}_{\kappa} \tag{48}$$

We now propose to develop a preconditioner for $\tilde{A}^{tri}_{\kappa} = \Phi^+ \tilde{A}^{tri}_{\kappa^\dagger} \Phi$ of the form:

$$\tilde{A}^{tri^{-1}}_{\kappa} \approx \Phi^+ \tilde{A}^{tri^{-1}}_{\kappa^\dagger} \Phi^{+^T} \tag{49}$$

where $\Phi^+ \equiv M^{tri^{-1}}_{\kappa} \Phi^T M^{tri}_{\kappa^\dagger}$ is a weighted pseudo-inverse of Φ, corresponding to the discrete L^2 projection operator from the artificially constructed tensor-product space to the polynomial space on the triangle. We note that the action of Φ^+ and Φ^{+^T} can be efficiently computed using the sum-factorization approach.

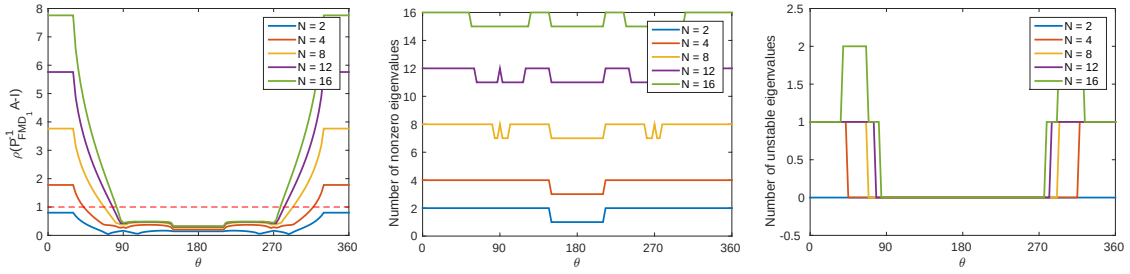
The design of our preconditioner is founded on the principle that the tensor-product form of $\tilde{A}^{tri}_{\kappa^\dagger}$ makes computing its inverse (or an approximation thereof) simpler than computing the inverse of $\tilde{A}^{tri}_{\kappa}$.

We write:

$$\tilde{A}^{tri}_{\kappa^\dagger} = |\mathbf{x}_\xi|(M_{1^\dagger} \otimes M_2)(G_{1^\dagger} \otimes I)\left((G^{-1}_{1^\dagger}\tilde{D}_{1^\dagger} \otimes I) + (I \otimes \tilde{D}_2)\right) \tag{50}$$

As discussed previously, the first term $|\mathbf{x}_\xi|(M_{1^\dagger} \otimes M_2)$ is simply the diagonal matrix of quadrature weights and can easily be inverted. Similarly, $(G_{1^\dagger} \otimes I)$ is a diagonal matrix which is easily inverted. The third matrix on the right-hand side of (50) is in separable tensor-product form such that we can now apply our fast-diagonalization method (FDM) or alternating-direction-implicit (ADI) schemes.

We first consider the case where $\tilde{A}^{tri}_{\kappa^\dagger}$ is inverted exactly and examine the performance of the proposed preconditioner $P^{tri^{-1}}_{FDM_1} = \Phi^{+^T}\tilde{A}^{tri^{-1}}_{\kappa^\dagger}\Phi^+$. Application of this preconditioner may be performed efficiently using the sum-factorization approach and the fast-diagonalization method. We evaluate the performance by examining the eigenvalue spectrum of the preconditioned operator $\left(P^{tri^{-1}}_{\kappa}A^{tri}_{\kappa} - I\right)$ as a function of θ, the angle of the flow relative to a reference equilateral triangle. We define our coordinate system such that $\theta = 0$ corresponds to flow into this reference element in a direction normal to the edge opposite the collapsed node. In Figure 4 we present the spectral radius of the preconditioned operator, the number of non-zero eigenvalues and the number of eigenvalues with magnitude greater than 1. As can be observed in Figure 4 the proposed preconditioner is unstable when the flow is towards the corner corresponding to the collapsed edge, with the largest eigenvalue approching $N/2$. However, we note that the majority of the eigenvalues of the preconditioned operator are identically zero, with either N or $N-1$ non-zero eigenvalues depending upon the angle of the flow. Additionally, for $N \leq 16$ there is only a single unstable eigenvalue, suggesting that we can apply a rank-1 perturbation to recover a stable preconditioner.



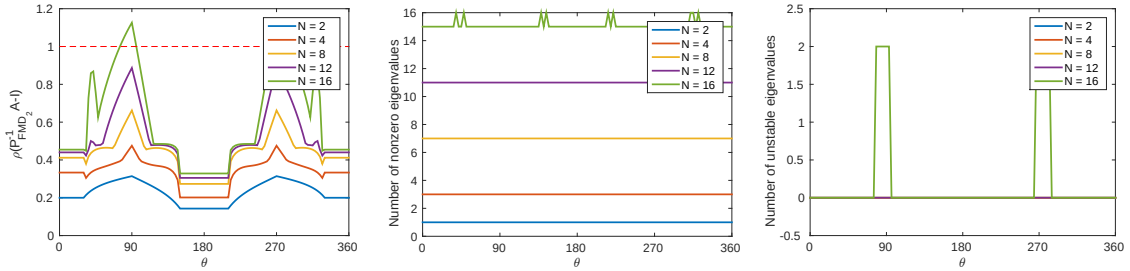
(a) Spectral Radius (b) Nonzero Eigenvalues (c) Unstable Eigenvalues

Figure 4. Spectral radius of $\left(P^{tri^{-1}}_{FDM_1}A^{tri}_{\kappa} - I\right)$

We next consider the preconditioner:

$$P^{tri^{-1}}_{FDM_2} = \left(I - \frac{P^{tri^{-1}}_{FDM_1}e_1e_2^T}{1+e_2^TP^{tri^{-1}}_{FDM_1}e_1}\right)P^{tri^{-1}}_{FDM_1} \tag{51}$$

which corresponds to the application of the Sherman-Morrison formula to correct for the inexact quadrature on the edges adjacent to the collapsed corner (though in this case $\tilde{A}^{tri^{-1}}_{\kappa}$ is replaced by our first preconditioner $P^{tri^{-1}}_{FDM_1}$). In Figure 5 we show the spectral radius and the number of non-zero eigenvalues, corresponding to this preconditioner. As desired, we recover a stable preconditioner for $N \leq 12$. Unfortunately, when the flow is into one of the edges next to the collapsed corner, the performance degrades significantly with increasing solution order and the preconditioner becomes unstable for $N = 16$. We note that the rank-1 perturbation applied does not, in general, correspond exactly to the largest eigenvalues of the preconditioned operator $P^{tri^{-1}}_{FDM_1}A^{tri}_{\kappa} - I$, and different rank-1 perturbations may be more successful. Alternatively, we could consider a rank-N perturbation which would result in an exact preconditioner. At this stage we have not explored further variants of a preconditioner based on the fast-diagonalization method for unstructured element shapes, however these preliminary results suggest that it may be possible to develop an efficient tensor product preconditioner by inverting $\tilde{A}^{tri}_{\kappa^\dagger}$.



(a) Spectral Radius (b) Nonzero Eigenvalues (c) Unstable Eigenvalues

Figure 5. Spectral radius of $\left(P_{FDM_2}^{tri^{-1}} A_\kappa^{tri} - I\right)$

Next, we develop an alternating-direction-implicit (ADI) preconditioner for A_κ^{tri}. We consider preconditioners of the form:

$$A_\kappa^{tri^{-1}} \approx \tilde{P}_{ADI}^{tri^{-1}} = \Phi^+ \tilde{P}_{ADI^\dagger}^{tri^{-1}} \Phi^{+^T} \tag{52}$$

where $\tilde{P}_{ADI^\dagger}^{tri^{-1}}$ is an ADI preconditioner for $\tilde{A}_{\kappa^\dagger}^{tri}$. First, consider a classical ADI preconditioner for $\tilde{A}_{\kappa^\dagger}^{tri}$ as given in (50). We introduce a pseudo-time τ and write the ADI preconditioner as:

$$\begin{aligned} \tilde{A}_{\kappa^\dagger}^{tri^{-1}} \approx \tilde{P}_{ADI_1^\dagger}^{tri^{-1}} &= \tau(I \otimes \tau\tilde{D}_2 + I)^{-1}(\tau G_{1^\dagger}^{-1}\tilde{D_{1^\dagger}} + I \otimes I)^{-1}(G_{1^\dagger} \otimes I)^{-1} \left(|\mathbf{x}_\xi|(M_{1^\dagger} \otimes M_2)\right)^{-1} \\ &= \tau(I \otimes \tau\tilde{D}_2 + I)^{-1}(\tau\tilde{D}_{1^\dagger} + G_{1^\dagger} \otimes I)^{-1} \left(|\mathbf{x}_\xi|(M_{1^\dagger} \otimes M_2)\right)^{-1} \end{aligned} \tag{53}$$

It can be easily shown that the spectral radius of the operator $(\tilde{P}_{ADI^\dagger}^{tri^{-1}} \tilde{A}_{\kappa^\dagger}^{tri} - I)$ is bounded by 1 for all positive τ,[16] while the minimum spectral radius is obtained with a pseudo-time such that the element CFL number $CFL_\kappa = N|\bar{a}|\tau = O(1)$. Unfortunately, these results do not translate directly to the spectral radius of $(\tilde{P}_{ADI}^{tri^{-1}} \tilde{A}_\kappa^{tri} - I)$ and we cannot bound this operator for all pseudo-time. Instead, we consider the limit as the pseudo-time goes to zero. For the quadrilateral, the ADI preconditioner recovers a scaled mass-matrix preconditioner in this limit. We wish our ADI-preconditioner for the triangular element to have the same feature. To this end we consider a second ADI preconditioner for $\tilde{A}_{\kappa^\dagger}^{tri}$ given by:

$$\tilde{P}_{ADI_2}^{tri^{-1}} = \tau(I \otimes \tau\tilde{D}_2 + I)^{-1}(\tau\tilde{D}_{1^\dagger} + I \otimes I)^{-1} \left(|\mathbf{x}_\xi|(M_{1^\dagger} \otimes M_2)\right)^{-1} \tag{54}$$

We note that the only difference between $\tilde{P}_{ADI_1^\dagger}^{tri^{-1}}$ and $\tilde{P}_{ADI_2}^{tri^{-1}}$ given in (53) and (54) respectively is the scaling of the pseudo-time term in the η_1 direction. In particular, $G_{1^\dagger}$ in (53) may be viewed as reducing the pseudo-time in the vicinity of the collapsed edge. Finally, we write our ADI preconditioner $\tilde{P}_{ADI^\dagger}$ as a combination of $\tilde{P}_{ADI_1^\dagger}^{tri^{-1}}$ and $\tilde{P}_{ADI_2}^{tri^{-1}}$

$$\tilde{P}_{ADI^\dagger}^{tri^{-1}} = \tau(I \otimes \tau\tilde{D}_2 + I)^{-1}(\tau\tilde{D}_{1^\dagger} + (\gamma G_{1^\dagger} + (1-\gamma)I) \otimes I)^{-1} \left(|\mathbf{x}_\xi|(M_{1^\dagger} \otimes M_2)\right)^{-1} \tag{55}$$

We note that the scaling of the pseudo-time in the η_1 direction is given by a linear combination governed by a single parameter $0 \leq \gamma \leq 1$. Based on numerical experiments we set γ as $\gamma = \frac{1}{4}\sqrt{N^2(\bar{a}_1^2 + \bar{a}_2^2)\tau^2}$, which generally gives satisfactory performance.

We now examine the performance of the proposed ADI preconditioner to solve the scalar advection problem on the triangle. We set τ such that $CFL_\kappa = N|\bar{a}|\tau = 1$ and compute the spectral radius of the preconditioned operator $(\tilde{P}_{ADI}^{tri^{-1}} \tilde{A}_\kappa^{tri} - I)$ as a function of the angle of the flow relative to a reference equilateral triangle.

As can be seen from Figure 6 the spectral radius of the proposed preconditioner appears bounded by 1. Additionally, the spectral radius grows slowly as a function of solution order N, suggesting that the proposed preconditioner may be used to eliminate the stiffness associated with high order.

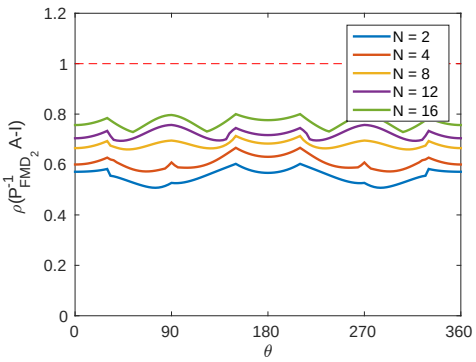


Figure 6. Spectral radius of $\left(P_{ADI}^{tri^{-1}} A_{\kappa}^{tri} - I\right)$

We now describe the extension of our ADI preconditioner to prisms, pyramid and tetrahedra. As with the triangle, we may form tensor product systems for the prism, pyramid and tetrahedra given by:

$$\begin{aligned}
\tilde{A}_{\kappa^\dagger}^{prism} &= |\mathbf{x}_\xi|(M_{1^\dagger} \otimes M_2 \otimes M_3)\left((\tilde{D}_{1^\dagger} \otimes I \otimes I) + (G_{1^\dagger} \otimes \tilde{D}_2 \otimes I) + (I \otimes I \otimes \tilde{D}_3)\right) \\
\tilde{A}_{\kappa^\dagger}^{pyramid} &= |\mathbf{x}_\xi|(M_{1^\ddagger} \otimes M_2 \otimes M_3)\left((\tilde{D}_{1^\ddagger} \otimes I \otimes I) + (G_{1^\ddagger} \otimes \tilde{D}_2 \otimes I) + (G_{1^\ddagger} \otimes I \otimes \tilde{D}_3)\right) \\
\tilde{A}_{\kappa^\dagger}^{tet} &= |\mathbf{x}_\xi|(M_{1^\ddagger} \otimes M_{2^\dagger} \otimes M_3)\left((\tilde{D}_{1^\ddagger} \otimes I \otimes I) + (G_{1^\ddagger} \otimes \tilde{D}_{2^\dagger} \otimes I) + (G_{1^\ddagger} \otimes G_{2^\dagger} \otimes \tilde{D}_3)\right)
\end{aligned}$$

where:

$$M_{1^\ddagger}[m,n] = \int_{\eta_1=-1}^{1} \left(\tfrac{1-\xi_1}{2}\right)^2 w_1 v_1 \tag{56}$$

$$D_{1^\ddagger}[m,n] = -\int_{\eta_1} \left(\left(\tfrac{1-\eta_1}{2}\right)^2 \bar{a}_1 w_{1,\eta_1} v_1\right) + \left[\left(\tfrac{1-\eta_1}{2}\right)^2 \bar{a}_1^+ w_1 v_1\right]_{\eta_1=-1} \tag{57}$$

where $M_{1^\ddagger}$ and $D_{1^\ddagger}$ are evaluated using a Gauss-Jacobi quadrature rule with $\alpha = 2$. The corresponding ADI preconditioners are given by:

$$\begin{aligned}
\tilde{P}_{\kappa^\dagger}^{prism^{-1}} &= \tau(I \otimes I \otimes \tau\tilde{D}_3 + I)^{-1}(I \otimes \tau\tilde{D}_2 + I \otimes I)^{-1} \times \\
&\quad (\tau\tilde{D}_{1^\dagger} + (\gamma G_{1^\dagger} + (1-\gamma)I) \otimes I \otimes I)^{-1} \left(|\mathbf{x}_\xi|(M_{1^\dagger} \otimes M_2 \otimes M_3)\right)^{-1} \qquad (58) \\
\tilde{P}_{\kappa^\dagger}^{pyramid^{-1}} &= \tau(I \otimes I \otimes \tau\tilde{D}_3 + I)^{-1}(I \otimes \tau\tilde{D}_2 + I \otimes I)^{-1} \times \\
&\quad (\tau\tilde{D}_{1^\ddagger} + (\gamma G_{1^\ddagger} + (1-\gamma)I) \otimes I \otimes I)^{-1} \left(|\mathbf{x}_\xi|(M_{1^\ddagger} \otimes M_2 \otimes M_3)\right)^{-1} \qquad (59) \\
\tilde{P}_{\kappa^\dagger}^{tet^{-1}} &= \tau(I \otimes I \otimes \tau\tilde{D}_3 + I)^{-1}(I \otimes \tau\tilde{D}_{2^\dagger} + (\gamma_2 G_{1^\dagger} + (1-\gamma_2)I) \otimes I)^{-1} \times \\
&\quad (\tau\tilde{D}_{1^\ddagger} + (\gamma G_{1^\ddagger} + (1-\gamma)I) \otimes I \otimes I)^{-1} \left(|\mathbf{x}_\xi|(M_{1^\ddagger} \otimes M_2 \otimes M_3)\right)^{-1} \qquad (60)
\end{aligned}$$

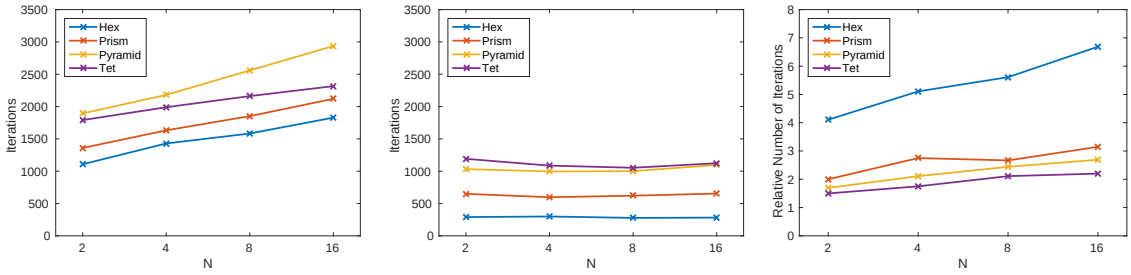
As with the triangle, γ scales the pseudo-time contribution in the η_1 and η_2 directions respectively. We set $\gamma = \sqrt{\frac{1}{4}N^2(\bar{a}_1^2 + \bar{a}_2^2)\tau^2}$ for the prism, $\gamma = \sqrt{\frac{1}{4}N^2(\bar{a}_1^2 + \bar{a}_2^2 + \bar{a}_3^2)\tau^2}$ for the pyramid and tetrahedra.

We now apply our ADI preconditioner to solve the scalar advection-diffusion problem, (27), in a unit box domain ($\Omega = [0,1]^3$). A forcing function is applied so that the exact solution is given by:

$$v = \sin(\pi x)\sin(\pi y)\sin(\pi z) \tag{61}$$

We solve the scalar advection problem using a velocity with unit magnitude and consider all flow angles $0 \leq \theta \leq 180°$ and $0 \leq \phi \leq 360°$, in 15° increments. The domain is partitioned uniformly to form a mesh with hexahedral elements, while each hexahedral element is then partitioned into either two prisms, six pyramids or six tetrahedra to form meshes of each element type. We solve the scalar advection problem using 2nd-, 4th-, 8th- and 16th- order spatial discretizations with approximately 64^3 degrees of freedom. We

evaluate the performance of our preconditioner in terms of the number of GMRES iterations required to converge the residual to 10^{-12}. Figure 7 reports the mean number of GMRES iterations over all flow angles using a mass-matrix preconditioner and the ADI preconditioner. Using the mass-matrix preconditioner, the number of iterations grows linearly with solution order, N, for a fixed number of degrees of freedom. The ADI preconditioner is able to reduce this stiffness, such that the number of iterations is roughly independent of solution order. Figure 7 also presents the median number of iterations using the mass-matrix preconditioner relative to the ADI preconditioner. The relative improvement is greatest for the hexadral mesh, while the relative improvement is much smaller using prisms, pyramids and tetrahedra. As a result, using a hexahedral mesh is approximately 2, 3.5 and 4 times more efficient than the corresponding prism, pyramid and tetrahedral meshes using similar numbers of degrees of freedom.



(a) Mass-Matrix Preconditioner (b) ADI Preconditioner (c) Relative Performance

Figure 7. Average number of GMRES iterations for scalar advection problem using mass-matrix and ADI preconditioners

IV. Preconditioning: Compressible Navier-Stokes equations

We now briefly discuss the extension of our ADI preconditioner to the unsteady compressible Euler equations. The full description of our diagonalized-ADI preconditioner for hexahedral elements appears in our previous papers.[6,8] As in our previous papers we consider the constant coefficient linearized Euler equations:

$$\boldsymbol{A}_0 \mathbf{v}_{,t} + \bar{\boldsymbol{A}} \nabla \mathbf{v} \quad = \quad 0. \tag{62}$$

Applying the discontinuous-Galerkin discretization, using the Roe flux[18] we obtain an elemental block Jacobian which corresponds to the following operator:

$$r(\boldsymbol{w}|_\kappa, \mathbf{v}|_\kappa) \quad = \quad -\int_{I^n}\int_\kappa (\boldsymbol{w}_{,t}\boldsymbol{A}_0\mathbf{v} + \nabla \boldsymbol{w} \cdot \bar{\boldsymbol{A}}\mathbf{v}) + \int_{I^n}\int_{\partial\kappa} \boldsymbol{w}\boldsymbol{A}_n^+\mathbf{v} + \int_\kappa \boldsymbol{w}(t_-^{n+1})\boldsymbol{A}_0\mathbf{v}(t_-^{n+1}). \tag{63}$$

As in the scalar case we will develop preconditioners based on applying an ADI scheme on the tensor-product space corresponding to the reference cube and then project the solution back to the reference prism, pyramid or tetrahedron. Again, we write the elemental block Jacobian in the following form:

$$\begin{aligned}
\frac{2}{\Delta t|\mathbf{x}_\xi|} r(\boldsymbol{w}|_\kappa, \mathbf{v}|_\kappa) \quad = \quad & + \int_\tau\int_{\eta_3}\int_{\eta_2}\left(-\int_{\eta_1} |\xi_\eta| \boldsymbol{w}_{,\eta_1}\tilde{\boldsymbol{A}}_1\mathbf{v} + \left[|\xi_\eta|\boldsymbol{w}\tilde{\boldsymbol{A}}_1^+\mathbf{v}\right]_{\eta_1=-1}^{\eta_1=1}\right) \\
& + \int_\tau\int_{\eta_3}\int_{\eta_1}\left(-\int_{\eta_2} |\xi_\eta| \boldsymbol{w}_{,\eta_2}\tilde{\boldsymbol{A}}_2\mathbf{v} + \left[|\xi_\eta|\boldsymbol{w}\tilde{\boldsymbol{A}}_2^+\mathbf{v}\right]_{\eta_2=-1}^{\eta_2=1}\right) \\
& + \int_\tau\int_{\eta_2}\int_{\eta_1}\left(-\int_{\eta_3} |\xi_\eta| \boldsymbol{w}_{,\eta_3}\tilde{\boldsymbol{A}}_3\mathbf{v} + \left[|\xi_\eta|\boldsymbol{w}\tilde{\boldsymbol{A}}_3^+\mathbf{v}\right]_{\eta_3=-1}^{\eta_3=1}\right) \\
& + \int_{\eta_3}\int_{\eta_2}\int_{\eta_1}\left(-\int_\tau |\xi_\eta| \boldsymbol{w}_{,\tau}\tilde{\boldsymbol{A}}_0\mathbf{v} + \left[|\xi_\eta|\boldsymbol{w}\tilde{\boldsymbol{A}}_0\mathbf{v}\right]^{\tau=1}\right),
\end{aligned} \tag{64}$$

where $\tilde{\boldsymbol{A}}_i = \eta_{i,x_j}\boldsymbol{A}_j$,

$$\tilde{\boldsymbol{A}}_i^+ = \left\{ \begin{array}{lll} \frac{1}{2}(\tilde{\boldsymbol{A}}_i + |\tilde{\boldsymbol{A}}_i|) & \text{at} & \eta_i = 1 \\ \frac{1}{2}(\tilde{\boldsymbol{A}}_i - |\tilde{\boldsymbol{A}}_i|) & \text{at} & \eta_i = -1 \end{array} \right\}, \tag{65}$$

while $\tilde{\boldsymbol{A}}_0 = \frac{2}{\Delta t}\boldsymbol{A}_0$. We again recognize that each line on the right-hand side of (64) corresponds to the discretization of a one-dimensional problem along an axis of the reference cube. However, unlike the scalar case these one-dimensional problems correspond to a 5×5 hyperbolic system. Following Pulliam and Chaussee[19] we diagonalized the flux Jacobians, $\tilde{\boldsymbol{A}}_i = \boldsymbol{R}_i \tilde{\boldsymbol{\Lambda}}_i \boldsymbol{R}_i^T$. Here $\boldsymbol{R}_i$ are the eigenvectors, while $\tilde{\boldsymbol{\Lambda}}_i$ is the matrix of eigenvalues of $\boldsymbol{A}_i$. We also note that $\boldsymbol{A}_0 = \boldsymbol{R}_0\boldsymbol{R}_0^T = \boldsymbol{R}_1\boldsymbol{R}_1^T = \boldsymbol{R}_2\boldsymbol{R}_2^T = \boldsymbol{R}_3\boldsymbol{R}_3^T$.[20] Thus we can write:

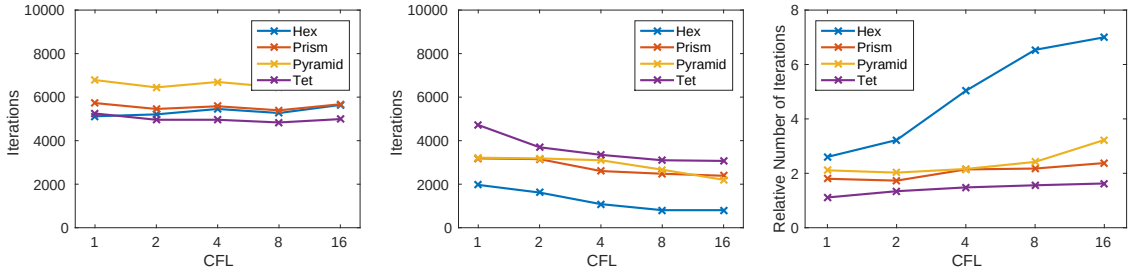
$$
\begin{aligned}
\frac{2}{\Delta t|\mathbf{x}_\xi|} r(\boldsymbol{w}|_\kappa, \mathbf{v}|_\kappa) \quad = \quad & + \int_\tau \int_{\eta_3} \int_{\eta_2} \left(- \int_{\eta_1} |\xi_\eta| \boldsymbol{w}_{,\eta_1} \boldsymbol{R}_1 \tilde{\boldsymbol{\Lambda}}_1 \boldsymbol{R}_1^T \mathbf{v} + \left[|\xi_\eta| \boldsymbol{w} \boldsymbol{R}_1 \tilde{\boldsymbol{\Lambda}}_1^+ \boldsymbol{R}_1^T \mathbf{v} \right]_{\eta_1=-1}^{\eta_1=1} \right) \\
& + \int_\tau \int_{\eta_3} \int_{\eta_1} \left(- \int_{\eta_2} |\xi_\eta| \boldsymbol{w}_{,\eta_2} \boldsymbol{R}_2 \tilde{\boldsymbol{\Lambda}}_2 \boldsymbol{R}_2^T \mathbf{v} + \left[|\xi_\eta| \boldsymbol{w} \boldsymbol{R}_2 \tilde{\boldsymbol{\Lambda}}_2^+ \boldsymbol{R}_2^T \mathbf{v} \right]_{\eta_2=-1}^{\eta_2=1} \right) \\
& + \int_\tau \int_{\eta_2} \int_{\eta_1} \left(- \int_{\eta_3} |\xi_\eta| \boldsymbol{w}_{,\eta_3} \boldsymbol{R}_3 \tilde{\boldsymbol{\Lambda}}_3 \boldsymbol{R}_3^T \mathbf{v} + \left[|\xi_\eta| \boldsymbol{w} \boldsymbol{R}_3 \tilde{\boldsymbol{\Lambda}}_3^+ \boldsymbol{R}_3^T \mathbf{v} \right]_{\eta_3=-1}^{\eta_3=1} \right) \\
& + \int_{\eta_3} \int_{\eta_2} \int_{\eta_1} \left(- \int_\tau |\xi_\eta| \boldsymbol{w}_{,\tau} \boldsymbol{R}_0 \boldsymbol{R}_0^T \mathbf{v} + \left[|\xi_\eta| \boldsymbol{w} \boldsymbol{R}_0 \boldsymbol{R}_0^T \mathbf{v} \right]^{\tau=1} \right), \qquad (66)
\end{aligned}
$$

Assuming that the eigenvectors of the flux Jacobian do not vary spatially we now recover 5 independent scalar advection problems in each coordinate direction. Following our derivation for the hexahedron,[6,8] we define our diagonalized-ADI scheme as given by the following sequence of steps:

1. Multiply by the inverse of the space-time element mass matrix
2. Transform to characteristic variables in the ξ_1-direction
3. Solve one-dimensional scalar systems along lines in the ξ_1-direction
4. Transform to characteristic variables in the ξ_2-direction
5. Solve one-dimensional scalar systems along lines in the ξ_2-direction
6. Transform to characteristic variables in the ξ_3-direction
7. Solve one-dimensional scalar systems along lines in the ξ_3-direction
8. Solve one-dimensional scalar systems along lines in the τ-direction
9. Transform back to entropy variables

In steps 2, 4, 6 and 9 variable transformations are performed locally at each quadrature point using the point-wise values for the state and geometry information. Similarly, the scalar systems solved in 3, 5, 7 and 8 correspond to variable-coefficient scalar-advection problems with advection velocity given by the eigenvalues averaged in a direction normal to the η_1-, η_2-, η_3- and τ- directions, respectively. The exact form of the scalar systems solved are similar to those for the scalar case and are included in the Appendix for completeness.

We now evaluate the performance of the ADI preconditioners to solve the compressible Navier-Stokes equations. We solve the Taylor-Green vortex problem described in detail in Section V.B. We solve the Taylor-Green vortex problem at a Reynolds number of $Re = 1600$ using our space-time discontinuous Galerkin method using an 8th-order spatial and a 4th-order temporal discretization. As in the case of the scalar advection-diffusion problem we use meshes composed of hexahedra, prisms, pyramids and tetrahedra using approximately 64^3 degrees of freedom. Figure 8 presents the average number of GMRES iterations required to converge the space-time residual to 10^{-14} using both mass-matrix and diagonalized-ADI preconditioners. We present the number of GMRES iterations as a function of the space-time CFL number, $CFL = \frac{|c|N_t\Delta t}{Nh}$ for a fixed time period. For the hexahedral mesh, the diagonalized-ADI preconditioner results in significantly fewer GMRES iterations, with increasing benefit for larger CFL. For meshes with prisms, pyramids and tetrahedra the performance of the diagonalized-ADI preconditioner is reduced. However, the preconditioner provides some improvement over the performance of the mass-matrix preconditioner.



(a) Mass-Matrix Preconditioner (b) ADI Preconditioner (c) Relative Performance

Figure 8. Average number of GMRES iterations for Taylor-Green vortex problem solved using 8th-order spatial and 4th-order temporal with mass-matrix and diagonalized-ADI preconditioners.

V. Numerical Results

V.A. Method of Manufactured Solutions

First, we verify the implementation of our numerical scheme using the method of manufactured solutions. We solve the steady compressible Navier-Stokes equations with a forcing term such that the exact solution is sinusoidal in the density, pressure and each component of the velocity vector:

$$\rho(x,y,z) = \rho_0 + \rho_x \cos(a_{\rho_x}\pi x) + \rho_y \cos(a_{\rho_y}\pi y) + \rho_z \cos(a_{\rho_z}\pi z) \quad (67)$$

$$u(x,y,z) = u_0 + u_x \cos(a_{u_x}\pi x) + u_y \cos(a_{u_y}\pi y) + u_z \cos(a_{u_z}\pi z) \quad (68)$$

$$v(x,y,z) = v_0 + v_x \cos(a_{v_x}\pi x) + v_y \cos(a_{v_y}\pi y) + v_z \cos(a_{v_z}\pi z) \quad (69)$$

$$w(x,y,z) = w_0 + w_x \cos(a_{w_x}\pi x) + w_y \cos(a_{w_y}\pi y) + w_z \cos(a_{w_z}\pi z) \quad (70)$$

$$p(x,y,z) = p_0 + p_x \cos(a_{p_x}\pi x) + p_y \cos(a_{p_y}\pi y) + p_z \cos(a_{p_z}\pi z) \quad (71)$$

The forcing term and exact solutions were provided using the MASA library.[21] A unit box domain ($\Omega = [0,1]^3$) is partitioned uniformly to form a mesh with hexahedral elements. Each hexahedral element is then partitioned into either two prisms, six pyramids or six tetrahedra to form meshes of each element type. Figure 9 presents the convergence of the L_2 error in the state and gradient as a function of the mesh size $h = DOF_{hex}^{-1/3}$. As expected, the L_2 error in the state converges at the formal order of accuracy, while the L_2 error in the gradient converges at a rate one order less.

The number of degrees of freedom for the corresponding prism, pyramid and tetrahedral meshes relative to the hexahedral mesh are:

$$DOF_{prism} = \frac{N^2(N+1)}{N^3} \quad (72)$$

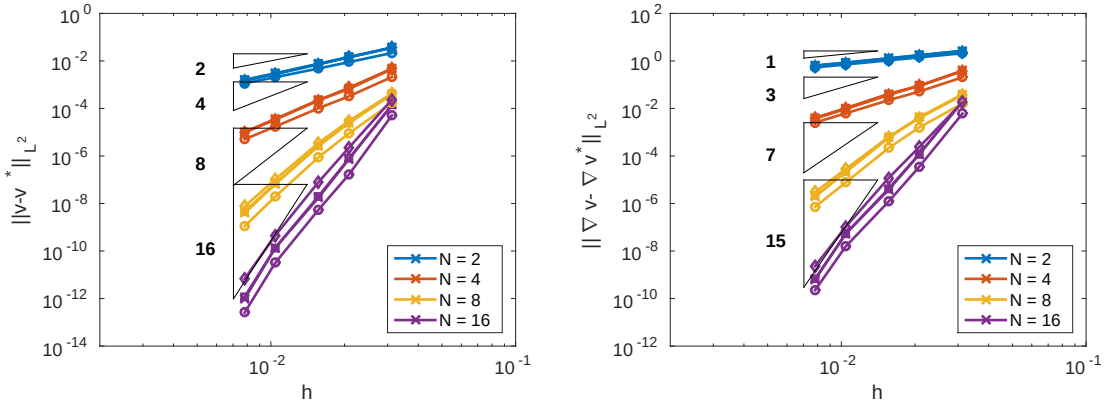
$$DOF_{pyramid} = \frac{N(N+1)(2N+1)}{N^3} \quad (73)$$

$$DOF_{tet} = \frac{N(N+1)(N+2)}{N^3} \quad (74)$$

At 2nd-order the hexahedral mesh is more efficient than the other mesh types since achieving the same error level requires equivalent prism, pyramid and tetrahedral meshes with roughly 3/2, 15/4 and 3 times as many degrees of freedom. For large polynomial orders the differences between the number of degrees of freedom for each mesh type decreases such that the same error level is achieved for all element types at similar number of degrees of freedom. The exception is the pyramid mesh which has an error level roughly half of the other mesh types. We note that our pyramid mesh is obtained by adding an additional node in the center of each hexahedron and splitting the hexahedron into 6 pyramids, allowing each hexahedra to be split independently. Alternatively, each hexahedron may be split into 3 pyramids, resulting in a mesh with half as many degrees of freedom. We suspect on such a mesh the results from the pyramid would fall directly on top of the other mesh types as well.

V.B. Taylor-Green Vortex

We now apply our higher-order discontinuous Galerkin method for the simulation of turbulent compressible flows. The Taylor-Green vortex evolution is used as a model problem for turbulent flow as it involves only



(a) State Error

(b) Gradient Error

Figure 9. Error versus mesh size for method of manufactured solution. (×-hex, □-prism, ◯-pyramid, ◊-tet)

periodic boundary conditions, no forcing and a simple initial condition. The flow is solved on an isotropic domain, which spans $[0, 2\pi L]$ in each coordinate direction. The initial conditions are given by:

$$u = V_0 \sin(x/L)\cos(y/L)\cos(z/L) \tag{75}$$

$$v = -V_0 \cos(x/L)\sin(y/L)\cos(z/L) \tag{76}$$

$$w = 0 \tag{77}$$

$$p = \rho_0 V_0^2 \left[\frac{1}{\gamma M_0^2} + \frac{1}{16}\left(\cos(2x/L) + \cos(2y/L)\right)\left(\cos(2z/L) + 2\right)\right] \tag{78}$$

where u, v and w are the components of the velocity in the x-, y- and z-directions, p is the pressure and ρ is the density. The Taylor-Green vortex flow is simulated using the compressible Navier-Stokes equations at Mach number $M_0 = 0.1$. The flow is initialized to be isothermal ($\frac{p}{\rho} = \frac{p_0}{\rho_0} = RT_0$). Simulations are performed at a Reynolds numbers $Re = \frac{\rho_0 V_0 L}{\mu} = 1600$.

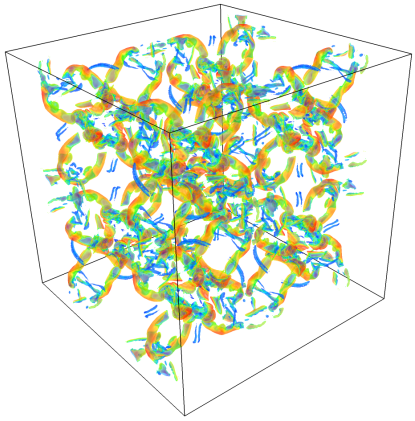


Figure 10. Iso-contours of vorticity magnitude at the instant of peak dissipation for the Taylor-Green vortex evolution at $M = 0.1$, Re = 1,600, computed using conservative variables, with 256^3 degrees of freedom.

Starting from the simple initial condition, the flow becomes turbulent through repeated vortex stretching leading to progressively smaller eddies, which are then dissipated to heat through the action of molecular viscosity. With increasing Reynolds number, progressively smaller structures appear. Figure 10 shows the iso-contours of vorticity at the instant of peak dissipation from a 16th-order solution at Re = 1,600.

For each simulation the temporal evolution of the kinetic energy

$$E_k = \frac{1}{\Omega}\int_\Omega \tfrac{1}{2}\rho \boldsymbol{V}\cdot\boldsymbol{V} d\Omega \tag{79}$$

is monitored. The evolution of the kinetic energy dissipation rate $\epsilon = -dE_k/dt$ was computed based on the data at the space-time quadrature points. We assess the quality of our numerical solutions by computing individual terms in the kinetic energy evolution equation. For compressible flow, the kinetic energy dissipation rate is given by the sum of three contributions $\epsilon = \epsilon_1 + \epsilon_2 + \epsilon_3 = -dE_k/dt$:

$$\epsilon_1 = \frac{1}{\Omega}\int_\Omega 2\mu \boldsymbol{S}:\boldsymbol{S} d\Omega \tag{80}$$

$$\epsilon_2 = \frac{1}{\Omega}\int_\Omega \lambda(\nabla\cdot\boldsymbol{V})^2 d\Omega \tag{81}$$

$$\epsilon_3 = -\frac{1}{\Omega}\int_\Omega p(\nabla\cdot\boldsymbol{V}) d\Omega \tag{82}$$

where $\boldsymbol{S} = \frac{1}{2}(\nabla\boldsymbol{V} + \nabla\boldsymbol{V}^T)$ is the strain rate tensor. Since the flow is nearly incompressible, we expect the dissipation due to the bulk viscosity, ϵ_2, and the pressure-dilatation term, ϵ_3, to be small. The kinetic energy dissipation rate is then approximately equal to $\epsilon \approx \epsilon_1$. However, for the compressible simulation this does not hold exactly.[7]

We perform simulations of the Taylor-Green vortex problem using meshes with hexahedra, prisms, pyramids and tetrahedra. We perform a mesh refinement study using our space-time DG method with 2nd- 4th- and 8th-order polynomials in space and 4th-order in time using 48, 64, 96 and 128 degrees of freedom in each coordinate direction. In Figure 11 we present the resolved viscous dissipation, ϵ_1, for different polynomial orders and element shapes. Reference data computed from an incompressible simulation using a spectral code on a 512^3 grid[22] is also presented. For 2nd-order schemes, less than half of the kinetic energy dissipation is resolved even on the finest grid considered. With increasing solution order, the results relative to the spectral data are significantly improved. At 8th-order the coarsest mesh considered resolves more of the viscous dissipation than using the finest mesh at 2nd-order.

In order to quantify these observations we evaluate the error in the computed kinetic energy dissipation rate:

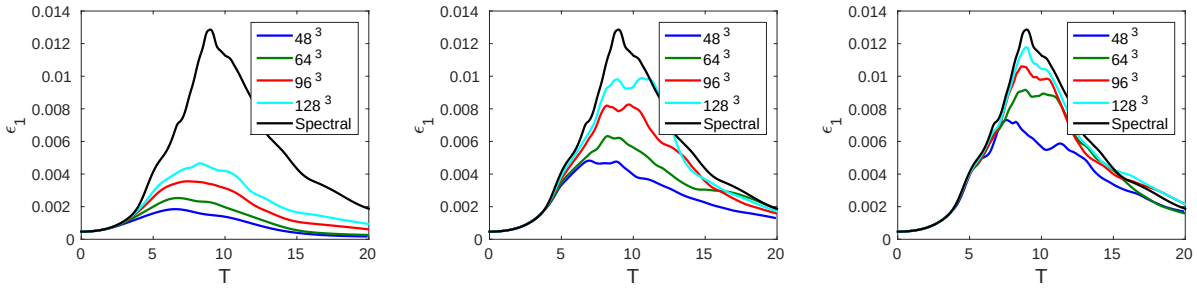
$$\text{Error} = \left| E_k(T) - E_k(0) + \int_0^T (\epsilon_1 + \epsilon_2 + \epsilon_3) dt \right| \tag{83}$$

Figure 12 presents the convergence of the error for 2nd-, 4th- and 8th- order spatial discretizations using meshes with hexahedra, prisms, pyramids and tetrahedra. For this test case we do not recover the formal convergence rate of our scheme, since at these mesh resolutions we are not in the asymptotic regime. However, even at these coarse mesh resolutions there is significant benefit to using the higher-order scheme as the 8th-order scheme has an error an order of magnitude less than the corresponding 2nd-order scheme using the same number of degrees of freedom.

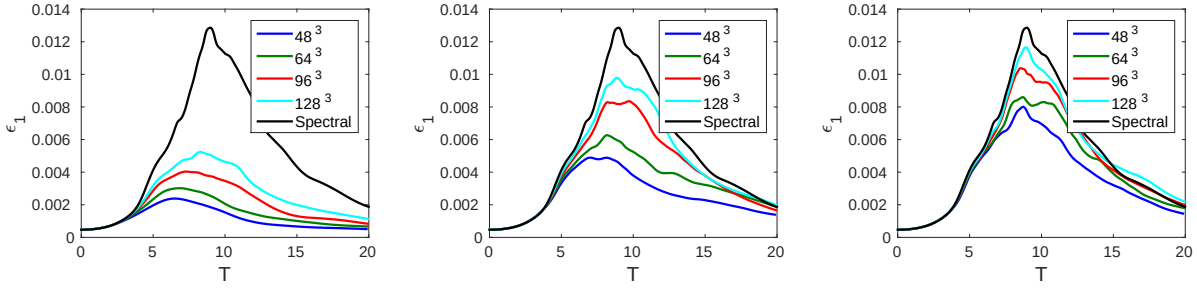
VI. Conclusions

We have extended our higher-order space-time discontinuous-Galerkin finite-element method to general element types. As with hexahedral meshes the use of a tensor-product formulation is key to achieving efficiency at high-order. We have presented extensions our ADI preconditioner to prisms, pyramids and tetrahedra by solving tensor-product problems on the reference cube. Numerical experiments demonstrated that the proposed ADI preconditioner is able to reduce the stiffness associated with high-order for scalar advection problems. The ADI preconditioner for the scalar-advection equations was then generalized to solve the compressible Euler and Navier-Stokes equations. Numerical results demonstrate the performance benefit of the proposed diagonalized-ADI relative to a simple mass-matrix preconditioner.

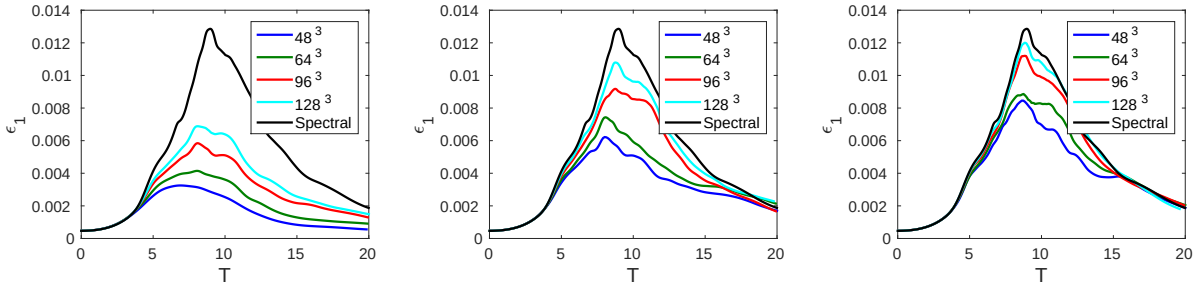
We have verified the formal accuracy of our numerical scheme up to 16th-order. Finally, we have applied our numerical scheme to simulate turbulent compressible flows. Numerical results demonstrate the efficiency of higher-order methods relative to a 2nd-order scheme.



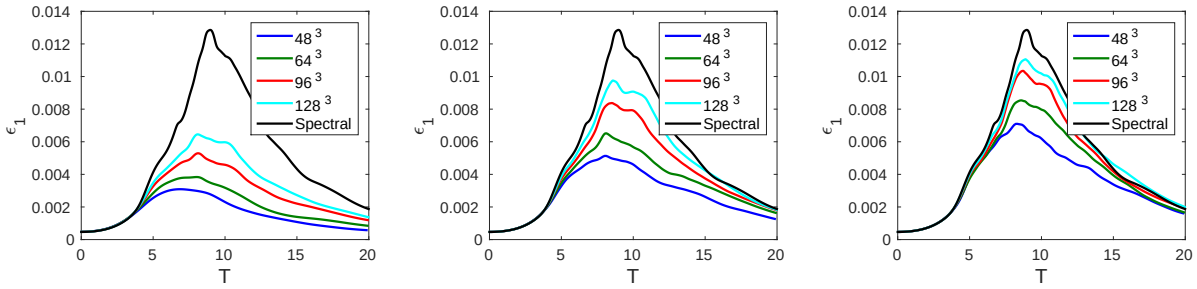
(a) $N = 2$, Hex (b) $N = 4$, Hex (c) $N = 8$, Hex



(d) $N = 2$, Prism (e) $N = 4$, Prism (f) $N = 8$, Prism



(g) $N = 2$, Pyramid (h) $N = 4$, Pyramid (i) $N = 8$, Pyramid



(j) $N = 2$, Tet (k) $N = 4$, Tet (l) $N = 8$, Tet

Figure 11. Evolution of the resolved viscous dissipation (ϵ_1) for the Taylor-Green vortex evolution at $M = 0.1$, $Re = 1{,}600$

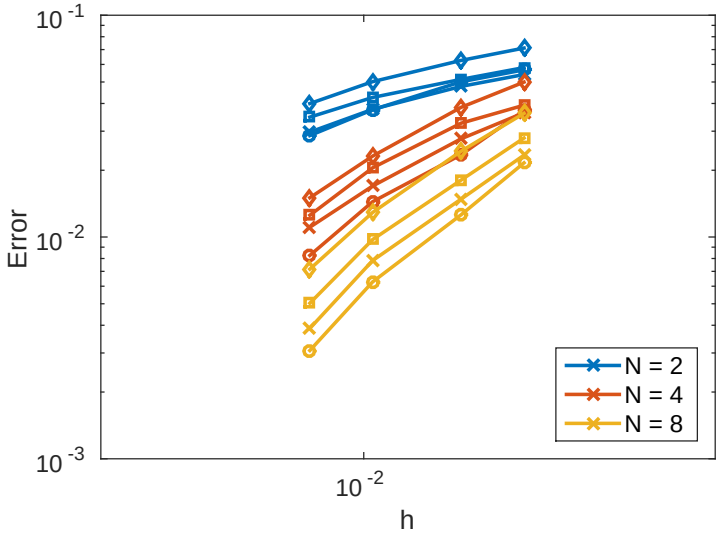


Figure 12. Error in kinetic energy evolution for Taylor-Green vortex problem at $M = 0.1$, $Re = 1{,}600$. ($\times$-hex, $\square$-prism, $\bigcirc$-pyramid, $\lozenge$-tet)

References

[1]Wang, Z., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H., Kroll, N., May, G., Persson, P.-O., van Leer, B., and Visbal, M., "High-Order CFD Methods: Current Status and Perspective," *International Journal for Numerical Methods in Fluids*, Vol. 72, 2013, pp. 811–845.

[2]Bassi, F. and Rebay, S., "GMRES discontinuous Galerkin solution of the compressible Navier-Stokes equations," *Discontinuous Galerkin Methods: Theory, Computation and Applications*, edited by K. Cockburn and Shu, Springer, Berlin, 2000, pp. 197–208.

[3]Fidkowski, K. J., Oliver, T. A., Lu, J., and Darmofal, D. L., "p-Multigrid solution of high-order discontiunous Galerkin discretizations of the compressible Navier-Stokes equations," *Journal of Computational Physics*, Vol. 207, No. 1, 2005, pp. 92–113.

[4]Persson, P.-O. and Peraire, J., "An efficient low memory implicit DG algorithm for time dependent problems," AIAA 2006-0113, 2006.

[5]Diosady, L. T. and Darmofal, D. L., "Preconditioning methods for discontinuous Galerkin solutions of the Navier-Stokes equations," *Journal of Computational Physics*, Vol. 228, 2009, pp. 3917–3935.

[6]Diosady, L. T. and Murman, S. M., "Design of a variational multiscale method for turbulent compressible flows," AIAA Paper 2013-2870, 2013.

[7]Diosady, L. T. and Murman, S. M., "DNS of flows over periodic hills using a discontinuous Galerkin spectral element method," AIAA Paper 2014-2784, 2014.

[8]Diosady, L. T. and Murman, S. M., "Tensor-Product Preconditioners for Higher-order Space-Time Discontinuous Galerkin Methods," 2014, under review.

[9]Diosady, L. T. and Murman, S. M., "Higher-Order Methods for Compressible Turbulent Flows Using Entropy Variables," AIAA Paper 2015-0294, 2015.

[10]Karniadakis, G. and Sherwin, S., *Spectral/hp element methods for CFD*, Oxford University Press, New York, NY, 1999.

[11]Kirby, R., Warburton, T., Lomtev, I., and Karniadakis, G., "A discontinuous Galerkin spectral/hp method on hybrid grids," *Applied Numerical Mathematics*, Vol. 33, No. 14, 2000, pp. 393 – 405.

[12]Kirby, R. M. and Karniadakis, G. E., "De-aliasing on non-uniform grids: algorithms and applications," *Journal of Computational Physics*, Vol. 191, 2003, pp. 249–264.

[13]Hughes, T. J. R., Franca, L., and Mallet, M., "A new finite element formulation for computational fluid dynamics: I Symmetric forms of the compressible Euler and Navier-Stokes equations and the second law of thermodynamics," Vol. 54, 1986, pp. 223–234.

[14]Ismail, F. and Roe, P. L., "Affordable, Entropy-consistent Euler flux functions II: entropy production at shocks," *J. Comput. Phys.*, Vol. 228, No. 15, Aug. 2009, pp. 5410–5436.

[15]Vos, P., Sherwin, S., and Kirby, R., "From h to p Efficiently: Implementing finite and spectral/hp element discretizations to achieve optimal performance at low and high order approximations." *Journal of Computational Physics*, Vol. 229, No. 13, 2010, pp. 5161–5181.

[16]Beam, R. and Warming, R., "An Implicit Factored Scheme for the Compressible Navier-Stokes Equations," *AIAA Journal*, Vol. 16, No. 4, 1978, pp. 393 – 402.

[17]Lynch, R. E., Rice, J. R., and Thomas, D. H., "Direct solution of partial difference equations by tensor product methods," *Numerische Mathematik*, Vol. 6, No. 1, 1964, pp. 185–199.

[18]Roe, P. L., "Approximate Riemann solvers, parameter vectors, and difference schemes," *Journal of Computational Physics*, Vol. 43, No. 2, 1981, pp. 357–372.

[19]Pulliam, T. and Chaussee, D., "A Diagonal Form of an Implicit Approximate-Factorization Algorithm," *Journal of Computational Physics*, Vol. 39, 1981, pp. 347–363.

[20]Barth, T. J., "Numerical Methods for Gasdynamic Systems on Unstructured Meshes," *An Introduction to Recent Developments in Theory and Numerics for Conservation Laws*, edited by D. Kroner, M. Olhberger, and C. Rohde, Springer-Verlag, 1999, pp. 195 – 282.

[21]Malaya, N., Estacio-Hiroms, K. C., Stogner, R. H., Schulz, K. W., Bauman, P. T., and Carey, G. F., "MASA: A Library for Verification Using Manufactured and Analytical Solutions," *Engineering with Computers*, 2012.

[22]van Ress, W., Leonard, A., Pullin, D., and Koumoutsakos, P., "A comparison of vortex and pseudo-spectral methods for the simulation of periodic vortical flows at high Reynolds number," *Journal of Computational Physics*, Vol. 230, 2011, pp. 2794–2805.

Appendix

For completeness we give the form of the scalar equations solved in each direction for the diagonalized-ADI scheme. Assuming that the eigenvectors $\boldsymbol{R}_i$ do not vary spatially, the elemental block Jacobian for the hex, prism, pyramid and tet may be written as:

$$\begin{aligned}
\tilde{A}^{hex}_{\kappa^\dagger} &\approx |\mathbf{x}_\xi|(M_1 \otimes M_2 \otimes M_3 \otimes M_0 \otimes I)\Big((I \otimes I \otimes I \otimes \tilde{D}_0 \otimes \boldsymbol{R}_0\Lambda_0\boldsymbol{R}_0^T) + (\tilde{D}_1 \otimes I \otimes I \otimes I \otimes \boldsymbol{R}_1\Lambda_1\boldsymbol{R}_1^T) \\
&\quad +(I \otimes \tilde{D}_2 \otimes I \otimes I \otimes \boldsymbol{R}_2\Lambda_2\boldsymbol{R}_2^T) + (I \otimes I \otimes \tilde{D}_3 \otimes \boldsymbol{R}_3 I \otimes \Lambda_3\boldsymbol{R}_3^T)\Big) \\
\tilde{A}^{prism}_{\kappa^\dagger} &\approx |\mathbf{x}_\xi|(M_{1^\dagger} \otimes M_2 \otimes M_3 \otimes M_0 \otimes I)\Big((I \otimes I \otimes I \otimes \tilde{D}_0 \otimes \boldsymbol{R}_0\Lambda_0\boldsymbol{R}_0^T) + (\tilde{D}_{1^\dagger} \otimes I \otimes I \otimes I \otimes \boldsymbol{R}_1\Lambda_1\boldsymbol{R}_1^T) \\
&\quad +(G_{1^\dagger} \otimes \tilde{D}_2 \otimes I \otimes I \otimes \boldsymbol{R}_2\Lambda_2\boldsymbol{R}_2^T) + (I \otimes I \otimes \tilde{D}_3 \otimes I \otimes \boldsymbol{R}_3\Lambda_3\boldsymbol{R}_3^T)\Big) \\
\tilde{A}^{pyramid}_{\kappa^\dagger} &\approx |\mathbf{x}_\xi|(M_{1^\ddagger} \otimes M_2 \otimes M_3 \otimes M_0 \otimes I)\Big((I \otimes I \otimes I \otimes \tilde{D}_0 \otimes \boldsymbol{R}_0\Lambda_0\boldsymbol{R}_0^T) + (\tilde{D}_{1^\ddagger} \otimes I \otimes I \otimes I \otimes \boldsymbol{R}_1\Lambda_1\boldsymbol{R}_1^T) \\
&\quad +(G_{1^\ddagger} \otimes \tilde{D}_2 \otimes I \otimes I \otimes \boldsymbol{R}_2\Lambda_2\boldsymbol{R}_2^T) + (G_{1^\ddagger} \otimes I \otimes \tilde{D}_3 \otimes I \otimes \boldsymbol{R}_3\Lambda_3\boldsymbol{R}_3^T)\Big) \\
\tilde{A}^{tet}_{\kappa^\dagger} &\approx |\mathbf{x}_\xi|(M_{1^\ddagger} \otimes M_{2^\dagger} \otimes M_3 \otimes M_0 \otimes I)\Big((I \otimes I \otimes I \otimes \tilde{D}_0 \otimes \boldsymbol{R}_0\Lambda_0\boldsymbol{R}_0^T) + (\tilde{D}_{1^\ddagger} \otimes I \otimes I \otimes I \otimes \boldsymbol{R}_1\Lambda_1\boldsymbol{R}_1^T) \\
&\quad +(G_{1^\ddagger} \otimes \tilde{D}_{2^\dagger} \otimes I \otimes I \otimes \boldsymbol{R}_2\Lambda_2\boldsymbol{R}_2^T) + (G_{1^\ddagger} \otimes G_2 \otimes \tilde{D}_3 \otimes I \otimes \boldsymbol{R}_3\Lambda_3\boldsymbol{R}_3^T)\Big) \qquad (84)
\end{aligned}$$

where $(\tilde{D}_i \otimes \Lambda_i)$ denotes the block diagonal matrix corresponding to the scalar advection problems in each coordinate direction such that (84) is the discrete set of equations corresponding to (64). We then define $\tilde{D}^*_i$, $\tilde{D}^*_{i^\dagger}$ and $\tilde{D}^*_{i^\ddagger}$ such that:

$$\begin{aligned}
(\tau\tilde{D}^*_i \otimes \Lambda_i) &\equiv (\tau\tilde{D}_i \otimes \Lambda_i) + (I \otimes I) \qquad (85)\\
(\tau\tilde{D}^*_{i^\dagger} \otimes \Lambda_i) &\equiv (\tau\tilde{D}_i \otimes \Lambda_i) + (I \otimes \gamma G_{i^\dagger} + (1-\gamma)I) \qquad (86)\\
(\tau\tilde{D}^*_{i^\ddagger} \otimes \Lambda_i) &\equiv (\tau\tilde{D}_i \otimes \Lambda_i) + (I \otimes \gamma G_{i^\ddagger} + (1-\gamma)I) \qquad (87)
\end{aligned}$$

We write the corresponding diagonalized ADI preconditioners as:

$$
\begin{aligned}
\tilde{P}_{\kappa^\dagger}^{hex^{-1}} &= \tau(I\otimes I\otimes I\otimes I\otimes \boldsymbol{R}_3^{-T})(I\otimes I\otimes I\otimes \tau\tilde{D}_0^*\otimes\tilde{\boldsymbol{\Lambda}}_0)^{-1}(I\otimes I\otimes \tau\tilde{D}_3^*\otimes I\otimes\tilde{\boldsymbol{\Lambda}}_3)^{-1} \\
&\quad (I\otimes I\otimes I\otimes I\otimes \boldsymbol{R}_3^{-1}\boldsymbol{R}_2)(I\otimes \tau\tilde{D}_2^*\otimes I\otimes I\otimes\tilde{\boldsymbol{\Lambda}}_2)^{-1}(I\otimes I\otimes I\otimes I\otimes \boldsymbol{R}_2^{-1}\boldsymbol{R}_1) \\
&\quad (\tau\tilde{D}_1^*\otimes I\otimes I\otimes I\otimes\tilde{\boldsymbol{\Lambda}}_1)^{-1}(I\otimes I\otimes I\otimes I\otimes \boldsymbol{R}_1^{-1})|\mathbf{x}_\xi|(M_1\otimes M_2\otimes M_3\otimes M_0\otimes I)^{-1} \\
\tilde{P}_{\kappa^\dagger}^{prism^{-1}} &= \tau(I\otimes I\otimes I\otimes I\otimes \boldsymbol{R}_3^{-T})(I\otimes I\otimes I\otimes \tau\tilde{D}_0^*\otimes\tilde{\boldsymbol{\Lambda}}_0)^{-1}(I\otimes I\otimes \tau\tilde{D}_3^*\otimes I\otimes\tilde{\boldsymbol{\Lambda}}_3)^{-1} \\
&\quad (I\otimes I\otimes I\otimes I\otimes \boldsymbol{R}_3^{-1}\boldsymbol{R}_2)(I\otimes \tau\tilde{D}_2^*\otimes I\otimes I\otimes\tilde{\boldsymbol{\Lambda}}_2)^{-1}(I\otimes I\otimes I\otimes I\otimes \boldsymbol{R}_2^{-1}\boldsymbol{R}_1) \\
&\quad (\tau\tilde{D}_{1^\dagger}^*\otimes I\otimes I\otimes I\otimes\tilde{\boldsymbol{\Lambda}}_1)^{-1}(I\otimes I\otimes I\otimes I\otimes \boldsymbol{R}_1^{-1})|\mathbf{x}_\xi|(M_{1^\dagger}\otimes M_2\otimes M_3\otimes M_0\otimes I)^{-1} \\
\tilde{P}_{\kappa^\dagger}^{pyramid^{-1}} &= \tau(I\otimes I\otimes I\otimes I\otimes \boldsymbol{R}_3^{-T})(I\otimes I\otimes I\otimes \tau\tilde{D}_0^*\otimes\tilde{\boldsymbol{\Lambda}}_0)^{-1}(I\otimes I\otimes \tau\tilde{D}_3^*\otimes I\otimes\tilde{\boldsymbol{\Lambda}}_3)^{-1} \\
&\quad (I\otimes I\otimes I\otimes I\otimes \boldsymbol{R}_3^{-1}\boldsymbol{R}_2)(I\otimes \tau\tilde{D}_2^*\otimes I\otimes I\otimes\tilde{\boldsymbol{\Lambda}}_2)^{-1}(I\otimes I\otimes I\otimes I\otimes \boldsymbol{R}_2^{-1}\boldsymbol{R}_1) \\
&\quad (\tau\tilde{D}_{1^\ddagger}^*\otimes I\otimes I\otimes I\otimes\tilde{\boldsymbol{\Lambda}}_1)^{-1}(I\otimes I\otimes I\otimes I\otimes \boldsymbol{R}_1^{-1})|\mathbf{x}_\xi|(M_{1^\ddagger}\otimes M_2\otimes M_3\otimes M_0\otimes I)^{-1} \\
\tilde{P}_{\kappa^\dagger}^{tet^{-1}} &= \tau(I\otimes I\otimes I\otimes I\otimes \boldsymbol{R}_3^{-T})(I\otimes I\otimes I\otimes \tau\tilde{D}_0^*\otimes\tilde{\boldsymbol{\Lambda}}_0)^{-1}(I\otimes I\otimes \tau\tilde{D}_3^*\otimes I\otimes\tilde{\boldsymbol{\Lambda}}_3)^{-1} \\
&\quad (I\otimes I\otimes I\otimes I\otimes \boldsymbol{R}_3^{-1}\boldsymbol{R}_2)(I\otimes \tau\tilde{D}_{2^\dagger}^*\otimes I\otimes I\otimes\tilde{\boldsymbol{\Lambda}}_2)^{-1}(I\otimes I\otimes I\otimes I\otimes \boldsymbol{R}_2^{-1}\boldsymbol{R}_1) \\
&\quad (\tau\tilde{D}_{1^\ddagger}^*\otimes I\otimes I\otimes I\otimes\tilde{\boldsymbol{\Lambda}}_1)^{-1}(I\otimes I\otimes I\otimes I\otimes \boldsymbol{R}_1^{-1})|\mathbf{x}_\xi|(M_{1^\ddagger}\otimes M_{2^\dagger}\otimes M_3\otimes M_0\otimes I)^{-1}
\end{aligned}
\tag{88}
$$

The application of $\boldsymbol{R}_3^{-T}$, $\boldsymbol{R}_3^{-1}\boldsymbol{R}_2$, $\boldsymbol{R}_2^{-1}\boldsymbol{R}_1$, and $\boldsymbol{R}_1^{-1}$ correspond to transformations to and from characteristic variables, which are performed locally at each quadrature point. Additionally, we note that $\boldsymbol{R}_3^{-1}\boldsymbol{R}_2$, $\boldsymbol{R}_2^{-1}\boldsymbol{R}_1$ depend only upon local geometry information.[19] Since, in general, the eigenvalues $\boldsymbol{\Lambda}_i$ and eigenvectors $\boldsymbol{R}_i$ vary spatially, the application of the diagonalized-ADI preconditioner cannot be written in the simplified form given in (88). Instead we define our diagonalized-ADI scheme as given by the sequence of steps described in Section IV. Each one-dimensional system then corresponds to a scalar advection problem with advection-velocity given by an eigenvalue averaged in the directions normal to η_1, η_2, η_3 and τ, respectively.